الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research

جامعة زيان عاشور بالجلفة
Ziane Achour University of Djelfa

كلية العلوم و التكنولوجيا
Faculty of Science and Technology

Department: Civil Engineering

**Order N°:……. /2022**

Defense authorization N°:……/2022

# DOCTORAL THESIS

Doctoral of science

Presented by

**BAKHTI Rachid**

With a view to obtaining the doctoral of science diploma

Branch: Civil engineering

***Topic***

## Build a computer Application for optimizing the damage calculations of concrete material

Supported, on 03 /07 / 2022, before the jury composed of:

| Last and first name | Grade | Institution of affiliation | Designation |
|---|---|---|---|
| Mr. BADAOUI Mohamed | MCA | University Ziane Achour, Djelfa | President |
| Mr. BENAHMED Baizid | MCA | University Ziane Achour, Djelfa | Supervisor |
| Mr. HARICHE Lazhar | MCA | University Ziane Achour, Djelfa | Examiner |
| Mr. DIF Fodil | MCA | University Ziane Achour, Djelfa | Examiner |
| Mr. ARBAOUI Ahcene | MCA | University Akli Mohand Oulhadj, Bouira | Examiner |
| Mr. ARIBI Chouaib | MCA | University Akli Mohand Oulhadj, Bouira | Examiner |

**Djelfa University, FST - 2022**

# ACKNOWLEDGEMENTS

The author is indebted to Dr. Baizid Benahamed for his interest, guidance, and provision of technical assistance during this research. Dr. Benahamed's commitment and assistance were limitless and this is greatly appreciated. Thanks are also extended to Dr. Laib Abdelghani from the University of Bouira and Dr. Mohanad Alfach from the University of London for their valuable time spent proofreading the published papers

I would like to express my very great appreciation to my thesis committee members, Dr. BADAOUI Mohamed, Dr. HARICHE Lazhar, Dr. DIF Fodil , Dr. ARBAOUI Ahcene, and Dr. ARIBI Chouaib for their valuable time.

**To my parents, wife, and my children (Abdelbasset, Abderraouf, and Nour)**

# ملخص

يعد تحليل العناصر المحدودة للخرسانة التالفة عملية معقدة للغاية من خلال حقيقة أن الخرسانة هي أكثر المواد تعقيدًا للنمذجة. ومع ذلك ، فإن الهدف الرئيسي من العمل الحالي هو تطوير رنامج حاسوبي مفتوح المصدر تحت مسمى "Concrete" باستخدام تقنية البرمجة الحديثة "OOP" لنمذجة عينات الخرسانة ذات الاشكال المكعبة والأسطوانية. تم برمجة مادة الخرسانية في البرنامج الحاسوبي المطور من خلال الشكل الثاني من النموذج المشهور (Plastic Damage Model PDM) للتعامل مع تدهور الصلابة المرنة الناتج عن الإجهاد بالإضافة إلى تأثيرات استعادة الصلابة تحت التحميل الدوري. في هذا العمل تم التخلص من عملية معايرة المعاملات المطلوبة في النموذج حيث تم اقتراح نهج رقمي جديد لتقدير كل من منحنيات اجهاد التوتر في حالتي الانضغاط والشد وكذا تطوير معامل تدهور الخرسانة في كلتا الحالتين. بالإضافة إلى القيم الافتراضية من لزاوية التمدد $\psi$ ، والانحراف $\varepsilon$ ، والنسبة $f_{b0}/f_{c0}$ ، والنسبة $Kc$ . تم اقتراح النهج الذي تم تطويره وفقًا لتوصيات Model Code

تم إنشاء الكود المطور في إطار Visual Studio 2019 (تم ترحيله إلى Visual Studio 2022) باستخدام لغة Vb.Net وتقنية WinForms لبناء المحرك وواجهة المستخدم الرسومية .GUI تم اختيار نموذج البرمجة OOP كأسلوب برمجي لتطوير برنامج الخرسانة. في نفس الاتجاه ، تم استخدام مكتبات OpenTk و Triangle.Net في البرنامج المطور من أجل تحسين عملية الرسم في كل من السرعة والجودة وإنشاء شبكات ثنائية الأبعاد للعينات الأسطوانية ، على التوالي.

كلمات مفتاحية: اللدونة الضرر للخرسانة ، طريقة العناصر المحدودة ، الهيكل الخرساني ، معاملات التلف

# Abstract

Finite element analysis of damaged concrete is a very complicated process by the fact that concrete is the most complex material to model in the analysis. However, the main aim of the present work is to provide an open-source finite element computer code under the name “Concrete” using the modern coding paradigm “Object-Oriented Programming” to model cubical and cylindrical concrete samples. The concrete material was implemented in the developed code through the second form of the famous constitutive low Plastic Damage Model (PDM) to handle the elastic stiffness degradation induced by the plastic straining in addition to the stiffness recovery effects under cyclic loading. The calibration process of the required parameters in the Damage plastic model was eliminated in the present work where a new numerical approach was suggested to auto-estimate each of; the stress-strain diagrams for the compressive and the tensile cases, in addition to the damage parameters evolutions for both cases. Also, default values from the literature were suggested for the dilation angle ψ, the eccentricity ε, the ratio $f_{b0}/f_{c0}$, and the ratio $Kc$. The developed approach was developed in accordance with the Model Code recommendations.

The developed code was built under visual studio 2019 (migrated to visual studio 2022) using the Vb.Net language and the WinForms technology to build the engine and the Graphical User Interface GUI; respectively. The Object-Oriented Programming paradigm was selected as a coding technique to develop Concrete software. In the same manner, the OpenTk and the Triangle.Net libraries were used in the developed software in order to improve the drawing process in speed and quality and generate 2D meshes for the cylindrical samples, respectively.

Key words: Concrete Damage Plasticity, finite element method, Concrete structure, damage parameters

# Table of Contents

# List of Figures

## Chapter I: The constitutive models for concrete: Overview



## Chapter II : Finite element implementation of Damage Plastic Model



## Chapter III: Description of the finite element computer code "Concrete v2.0.0"

## Chapter IV: Investigation for the outcomes of Concrete v2.0.0

# List of Tables

## Chapter I: The constitutive models for concrete: Overview



## Chapter II : Finite element implementation of Damage Plastic Model



## Chapter III: Description of the finite element computer code "Concrete v2.0.0"



## Chapter IV: Investigation for the outcomes of Concrete v2.0.0

# Introduction

## Introduction

Concrete is one of the most important materials in construction, which present all around us, in concrete bridges, dams, buildings. In fact , concrete is the most widely used building material in the world, the global world production of this material in 2021 exceeding 10000 million cubic meters [1] and the global world cement production capacity reaching 4470.3 million tons in 2018 [2]. Actually, the use of concrete in construction worldwide is twice more than any other building material, including wood, steel, plastic, and aluminum. Hence, it is very important to be able to model the concrete material properly using the finite element method. However, concrete is the most complex material to model in the analysis. Many research efforts were conducted on understanding the behavior of concrete, and numerous papers were published on modeling concrete for numerical simulations. When subjected to very small stresses, the material behaves linearly, and elastically, but beyond certain threshold values, the cracking in tension will be observed, with thereafter tension softening, crushing in compression, and all in a highly nonlinear manner. This complex comportment of concrete must be simulated in general nonlinear analyses, for dynamic/ cyclic or static loading. To capture this behavior for analysis purposes, many researchers have pursued a phenomenological approach in which the material behavior of concrete is represented through multiple constitutive models available in the literature. These models can be categorized as follow; the empirical models [3]–[5], the linear and the nonlinear elastic models [6]–[12], the plastic models [13]–[17], the fracture models [18], [19], the endochronic models [20]–[22], and the damage models [23]–[29]. In civil engineering, a large number of FE programs were developed using various constitutive models in order to identify the concrete behavior. These programs are mainly created by using the Procedural Oriented Programming (POP) technique where each program collects a set of functions and subroutines. The reliability of the POP paradigm in processing complex algorithms was demonstrated. However, this approach does not address quality issues and program design. Programs created via POP paradigm have intricate control strategies, and internal data representation, therefore, these codes face difficulty in their maintenance and update process. In fact, the development of finite element software is a very complicated process that takes long and hard work to provide a commercial code. Hence, the main goal in developing a new FE Software is to keep maintenance, modification, and updating as simple as possible, which the Object-Oriented Programming paradigm (OOP) can easily offer

One of the most used models for concrete behavior was developed by Lubliner et al [26] under the name Plastic Damage Model (PDM). On one hand, the PDM suggested by Lubliner considers the elastic stiffness degradation caused by the plastic straining (for both cases tension and compression),

but unfortunately on the other hand, the suggested form cannot address the cyclic/dynamic loading. To overcome this issue, Lee and Fenves [28] developed a second form of the PDM where several modifications in the initial form were suggested. The initial form of the PDM was implemented in a standard finite element program for the first time by Oller et al [30] using the Procedural Oriented Programming (POP) technique. Later, Lee and Fenves [31] suggested a return-mapping algorithm to implement the recent form of the PDM. Also, Ahmed et al [32] implemented the second form of the PDM using a novel stress decomposition. The PDM was implemented in the finite element code ABAQUS under the name Concrete Damaged Plasticity Model (CDPM). The use of CDPM in ABAQUS requires several parameters which are: the stress-inelastic strain diagrams and the damage parameters evolution for compression and tension cases, the ratio $K_c$, the eccentricity $\varepsilon$, the ratio $f_{b0}/f_{c0}$, the dilation angle $\psi$ , and the viscosity parameter. The calibration of these parameters with experimental data complicates the use of the CDPM in ABAQUS which deeply diminishes its efficiency.

The aim of this work is to develop a computer code under the name “Concrete” to model cylindrical and cubical concrete samples using the PDM as a constitutive model with minimum numbers of required parameters. This work provides in the first chapter a full overview of the constitutive models available in the literature, where a detailed description of several categories of models was provided namely; the empirical models, the linear and the nonlinear elastic models, the plastic models, the fracture models, the endochronic models, and the Damage models.

The second chapter of this thesis provides an overview of two well know finite element implementations of the PDM which were developed by Oller and Lee respectively. A new numerical methodology was delivered in this chapter to compute the compressive and the tensile stress-strain curves and the damage parameters evolutions in accordance with the Model Code recommendations [33]. The developed numerical approach is based mainly on Lubliner formulas [26] and Alfarah formulas[23]. Furthermore, default values were suggested for each; the ratio $K_c$, the eccentricity $\varepsilon$, the ratio $f_{b0}/f_{c0}$, and the dilation angle $\psi$ . A full description of the proposed finite element implementation of the PDM was delivered in this chapter where a new closed-form solution of the plastic multiplier was provided, in addition to the derivative of each; the yield function with respect of stresses, the derivative of the yield function with respect of inelastic strain, and the derivative of the potential function with respect of stresses.

The third chapter presents the user manual of the second version of “Concrete”, in addition to the used coding paradigm, and the used libraries. Also, a detailed description of the PDM class was provided in this chapter including the required fields, functions, and subroutines, in addition to the used algorithms. Finally, the last chapter presents a complete validation of the developed computer code “Concrete v2.0.0” by comparing its outcomes (stress-strain curves) with experimental evidence and with analytical solutions. Likewise, the suggested methodology for computing the stresses and the damage parameters evolution was examined through a comparative study with solutions from the literature. Also, the mesh sensitivity was examined in this chapter

# Chapter I : The constitutive models for concrete: Overview

## I.1 Introduction

Concrete is one of the most used materials in the construction field. In fact, the use of concrete is twice time more than the use of all other construction materials. Thus, the pervasive use of this material dictates a thorough understanding of the real behavior of concrete. As demonstrated in the literature, the concrete stress-strain behavior under uniaxial compression or tension loading is nonlinear. Therefore, simulating the concrete behavior with linear models in the numerical modeling using FEM leads to inaccurate results which makes the non-linear models an unavoidable key to minimize the error margin. An accurate solution for a concrete structural problem depends mainly on the used constitutive model that must be describe the real behavior of concrete. In fact, the concrete behavior is very complicated since it is not the same in the compression and the tension cases. It can be said that perhaps impossible to find any phenomenological approach can describe all the possible variations of concrete characteristics.

Several constitutive models were developed in the few last decades in order to simulate the complex behavior of concrete. These models are classified according to Babu et al [34] in seven categories. The first one is the empirical models [3]–[5] where the constitutive equations are mainly developed basing on the outcomes of the experimental tests. The second category based on the Hook's low and baptized the linear elastic models [6]–[8]. The third is called the nonlinear elastic models [9]–[12] which are mainly characterized by the nonlinear stress-strain relationship. The forth category adopt the plasticity theory to describe the concrete behavior [13]–[17]. The category number five called the fracture models [18], [19] where the models are based on the concept of propagation of microcracks. In the aim of eliminating the yield function complexity, anther category of models was developed under the name endochronic models [20]–[22], but in the few last decades the development of this type of models has no more supported by the scientific community. Finally the last category of models is the damage models [23]–[29] where the constitutive equations consider the damage parameters evolution and the loss of cohesion.

One of the most innovative models for concrete was developed by Lubliner et al [26] under the name Plastic Damage Model (PDM), and upgraded by Lee and Fenves [28] to overcome the inability of handling the cyclic / dynamic loading. The recent form of this model was exploited in several research works. For instance, Javanmardi and Maheri [35] used Lee and Fenves form to predict the crack propagation paths. Also, Bilal et al [32] suggested a novel stress decomposition using the work of Lee and Fenves. Similarly, Poliotti and Bairán [36] developed a new constitutive plastic-damage model with evolutive dilatancy. The second form of this model was implemented in the finite element

code ABAQUS in the late of '90s under the name Concrete Damaged Plasticity Model (CDPM). The package software ABAQUS has been widely used in the numerical modeling of concrete using CDPM. For instance Silva et al [37] used CDPM to simulate the concrete damage. Likewise, Ren et al [38] used the CDPM in the numerical simulation of prestressed precast concrete bridge deck panels. Furthermore, Othman and Marzouk [39] used it to simulate ultra-high performance fibre reinforced concrete material under impact loading rates at different damage stages. The CDPM has been widely used in the concrete numerical modeling where the following parameters are required:

- The stress-inelastic strain diagram for compression.
- The stress-inelastic strain diagram for tension.
- The ratio of the second stress invariants on tensile and compressive meridians ( $Kc$ ).
- The eccentricity (ε).
- The ratio of biaxial compressive yield stress to uniaxial compressive yield stress. ( $f_{b0}/f_{c0}$ )
- The dilation angle in the p-q plane (ψ).
- The viscosity parameter.
- The compressive damage parameter evolution.
- The tensile damage parameter evolution.

The main inconvenience of CDPM is that the outcomes are strongly depend on these parameters, values. Thus, all parameters must be calibrated with experimental tests. Furthermore, both of the high complexity and sensitivity of the calibration process deeply diminishes the CDPM efficiency.

This chapter presents a full description for each of the linear and nonlinear elastic models, Plasticity models, endochronic models, empirical models, and the damage models

## I.2 Linear and nonlinear elastic models

The linear elastic model is the simplest constitutive model for modeling the concrete behavior. This constitutive law presumes that there is a linear relationship between stress and strain governed by the Hooke law and the stress tensor is generated only by the elastic strain (the plastic part equal to zero). In the linear elastic model, concrete is treated as linear elastic until it reaches ultimate strength. For concrete under tension, since the failure strength is small, linear elastic model is quite accurate and sufficient to predict the concrete behavior till failure. Linear elastic stress-strain relation can be written as:

$$\sigma = D.\varepsilon \tag{1.1}$$

Where D represents the constitutive matrix given by:

| 1D | 2D | 3D |
|---|---|---|
| $D=E$ | $D=\frac{E}{1-\upsilon}\begin{bmatrix}1 & \upsilon & 0\\ \upsilon & 1 & 0\\ 0 & 0 & \frac{1-\upsilon}{2}\end{bmatrix}$ | $D=\frac{E}{(1+\upsilon)(1-2\upsilon)}\begin{bmatrix}1-\upsilon & \upsilon & \upsilon & 0 & 0 & 0\\ \upsilon & 1-\upsilon & \upsilon & 0 & 0 & 0\\ \upsilon & \upsilon & 1-\upsilon & 0 & 0 & 0\\ 0 & 0 & 0 & \frac{1-2\upsilon}{2} & 0 & 0\\ 0 & 0 & 0 & 0 & \frac{1-2\upsilon}{2} & 0\\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\upsilon}{2}\end{bmatrix}$ |

Both linear and nonlinear elastic materials are characterized by the elastically return to the "unloaded" state after loading but the main difference between them is the stress-strain relationship is linear for the linear elastic materials and more complex in the nonlinear case (the stress-strain relationship is nonlinear). In fact the nonlinear elastic models represent an extension of linear elastic models. Since the stress-strain relationship for concrete is nonlinear, the linear elastic model cannot provide accurate outcomes and the error margin becomes significant particularly for the compressive case. For the nonlinear case, two approaches can be used to model the nonlinear elastic materials. The first approach is the tangential stiffness method where the stiffness degradation is considered as failure is approached and the global stiffness matrix must be evaluated for each iteration. The second approach called the constant stiffness method in which the global stiffness matrix is evaluated once only in the iteration beginning. In this approach, the non-linearity is considered by iteratively modifying the loads vector.

Several criteria are available in the literature to define the failure for linear and nonlinear elastic models. The most reputed ones are defined through several independent control parameters (one until five parameters). Menetrey and Willam [40] developed one of the most sophisticated failure criteria basing on the work of Hoek and Brown with some modifications. The failure criteria suggested by Menetrey and Willam takes the following form:

$$f(\xi,p,\theta)=\left[\sqrt{1.5}\frac{p}{f_c^{'}}\right]+m\left[\frac{\ddot{p}}{\sqrt{6}f_c^{'}}r(\theta,e)+\frac{\xi}{\sqrt{3}f_c^{'}}\right]-c=0 \qquad (1.2)$$

Where $\xi=\frac{I_1}{\sqrt{3}}$ is the hydrostatic stress invariant. $p$ is the deviatoric stress invariant. $\theta$ is the deviatoric polar angle and $r(\theta,e)$ is an elliptic function.

## I.3 Plasticity models

Plasticity models were largely used in the few last decades to simulate the concrete behavior using the finite element method. In this category of models, the incremental strain can be divided to elastic and plastic parts. Two cases can be raised, the first one is when the plastic part of the incremental strain takes a value equal to zero which means that the stress state lies inside/ or in the yield surface (typically corresponds to purely elastic response which detected when the yield function takes a negative/zero value). The second case correspond the stress state is inaccessible (outside the yield surface which detected when the yield function takes a positive value).

In the literature, two kinds of the yield functions were developed. The first one is the yield functions that influenced by the hydrostatic pressure and the second one is that completely independent of the hydrostatic pressure. Bridgman demonstrated that hydrostatic pressure has a negligible effect on the yield point for large number of materials such as concrete which has a behavior influenced by the effect of hydrostatic pressure. For instance, table 1.1 summarizes various well-known yield functions from both kinds. A number of models were developed specifically for concrete such as Menetrey and Willam [40] and Ottosen [41] where several modifications into the plasticity theory were suggested to compute the strain and the stress.

**Table 1.1:** Types of the yield functions

| Type | Yield function |
|---|---|
| Pressure independent | - Tresca<br>- Von-Mises<br>- Rankine |
| Pressure dependent | - Mohr-Coulomb<br>- Drucker-Prager<br>- Mises-Schleicher |

As explained previously, in the plasticity theory the total strain increment is the sum of the elastic part and the plastic part of strain increment, so the total strain increment can be written as:

$$\{\Delta \varepsilon\} = \left\{\Delta \varepsilon^{e}\right\} + \left\{\Delta \varepsilon^{p}\right\} \tag{1.3}$$

Where the plastic component can be evaluated according to flow rule by:

$$\left\{\Delta \varepsilon^{p}\right\} = d\lambda \frac{\partial G}{\partial \sigma} \tag{1.4}$$

With $d\lambda$ is the plastic multiplier and G is the potential function. The incremental stress can be evaluated from the incremental elastic strain according to following relationship:

$$\{\Delta\sigma\} = [D]\left\{\Delta\varepsilon^{e}\right\} \tag{1.5}$$

Where $[D]$ represent the elastic constitutive matrix.

By using Eqs (1.3),(1.4) , and (1.5), the incremental stress can be evaluated by:

$$\{\Delta\sigma\} = [D]\{\Delta\varepsilon\} - d\lambda[D]\frac{\partial G}{\partial\sigma} \tag{1.6}$$

For concrete material, Han and Chen [42], Dvorkin et al. [43] suggested to use the Drucker-Prager yield function as potential function in order to evaluate the plastic strain, the potential function suggested by Han and Chen [42], Dvorkin et al. [43] takes the following form:

$$G = \alpha I_1 + \sqrt{J_2} + C \tag{1.7}$$

Where:

| | |
|---|---|
| $\alpha$ | Coefficient that can be calculated by<br>$\alpha = \frac{1}{\sqrt{3-(1-\frac{\varepsilon^p}{\varepsilon_v^p})}}$<br>with $\varepsilon_v^p$ is the volumetric part of the plastic strain that equal to the second invariant of stress tensor |
| $I_1$ | first invariant of stress tensor |
| $J_2$ | second invariants of deviatoric stress tensor |
| C | constant |

Vermeer and de Borst [44] used the constitutive model of Mohr-Coulomb to provide a new potential function where the main modification is substituting the internal friction angle $\phi$ by the dilatancy angle $\psi$ . The potential function suggested by Vermeer and de Borst [44] is given by:

$$G = \frac{I_1}{3}\sin\psi + \sqrt{J_2}\left(\cos\theta - \frac{\sin\theta\sin\psi}{\sqrt{3}}\right) \tag{1.8}$$

Where:

| | |
|---|---|
| $\theta$ | Lode angle |
| $\psi$ | Dilatancy angel |

## I.4 Endochronic theory of inelasticity

The main challenge in the plasticity models is finding the suitable correlations of the yield and the potential functions. To overcome this issue, another type of models was developed which did not require the existence of the yield function. This type of models is based on the concept of endochronic (or intrinsic) time calculated based on strain or stress and can be used to evaluate the structure damage. This model was initiated in 1971 by Valanis [45] to handle metals material and extended in 1976 by Bazant and Bhat [22] to address concrete material. In 1986 Reddy and Gopal [46] suggest a new form of this model in order to simulate the fibre reinforced concrete structures.

The major inconvenience of the Endochronic model is its complexity and a large number of the required parameters for the development and the application of this model, which significantly limit its use. Consequently, in the few last decades, this model has no more supported by the scientific community.

The intrinsic time ξ introduced by endochronic theory is given by:

$$\xi = \int_0^{\zeta} \frac{d\zeta}{f(\zeta)} \tag{1.9}$$

Where $f(\zeta)$ is the history-dependent material function takes values greater than zero and $d\zeta > 0$.

The typical constitutive equation for endochronic theory with pseudo-time measure ξ takes the following formula:

$$\sigma_{ij} = \int_0^{\xi} E_{ijkl}(\xi - \xi') \frac{\partial \varepsilon_{kl}}{\partial \varepsilon'} \tag{1.10}$$

## I.5 Empirical models

In this category, the constitutive law is obtained from a series of experiments tests, where the experimental outcomes are used to develop functions described the material behavior. The big challenge in developing an empirical model is how to obtain the experimental data, especially after the peak point. In fact, the test process for compression and tension cases is very complicated and requires sophisticated equipment. Most testing machines use increasing loads instead of deformations to record the stress-strain curves for the standard compression/tensile tests, resulting in uncontrolled sudden failure after peak load. The experimental stress-strain curves are evaluated following the specimen's shape and height-to-width ratio, which differ from a country to another according to the adopted standard. The specimen shape can significantly affect the stress-strain evolution, in particular the ultimate strain in compression, relevant peak load and the value of the descending arm of strain.

According to the European standard's, the concrete stress-strain curves both in compression and tension are evaluated based on a cylindrical specimen with a diameter-to-height ratio equal to 1/2, e.g. specimens with dimensions 160x320 mm

Several research tried to provide correlations between the stress and the strain for both cases compression and tension. Desayi and Krishan [47] suggested the following formula to describe the stress strain relationship in the compression case:

$$\sigma = \frac{E\varepsilon}{1+(\frac{\varepsilon}{\varepsilon_p})^2} \tag{1.11}$$

Where σ, ε are stress and strain tensors, $E$ is Young's modulus, $\varepsilon_p$ is strain at peak stress.

In the same manner, Saenz[5] proposed the next correlation:

$$\sigma = \frac{E\varepsilon}{1+(\frac{E}{E_p}-2)(\frac{\varepsilon}{\varepsilon_p})+(\frac{\varepsilon}{\varepsilon_p})^2} \tag{1.12}$$

Where $E_p$ is Young's modulus at peak stress

Also, Smith and Young [48] suggested the following formula:

$$\sigma = E\varepsilon\varepsilon^{\frac{-\varepsilon}{\varepsilon_p}} \tag{1.13}$$

Furthermore, Richard and Abbott [49] proposed a three parameter stress-strain relation given as:

$$\sigma = \frac{E_1\varepsilon}{(1+(\frac{E_1\varepsilon}{\sigma_0})^n)^{\frac{1}{n}}} + E_p\varepsilon \tag{1.14}$$

Where $E_p$ is plastic modulus, $\sigma_0$ is a reference plastic stress, $E_1 = E - E_p$ and $n$ is a shape parameter of stress-strain curve

Another simple form suggested by Mohamad Ali et al [50] is similar to the form proposed by Carreira and Chul [51] given by:

$$\frac{\sigma}{\sigma_0} = \frac{R\frac{\varepsilon}{\varepsilon_0}}{1+(R-1)(\frac{\varepsilon}{\varepsilon_0})^{\beta}} \tag{1.15}$$

Where $\beta = \frac{R}{R-1}$

And R is a material parameter depends on the shape of the stress-strain curve takes value equal to 1.9 according to Mohamad Ali et al [50]. The material parameter can be estimated by

$$R = \frac{E_c}{E_0}. \tag{1.16}$$

With $E_0$ is the concrete elasticity modulus and $E_c$ represent max. stress / strain at max. stress

Carreira and Chu [52] proposed a stress-strain relation for reinforced concrete in tension given by:

$$\frac{\sigma_t}{\sigma_t^{'}} = \frac{\beta\frac{\varepsilon}{\varepsilon^{'}}}{\beta-1+(\frac{\varepsilon}{\varepsilon_t^{'}})^{\beta}} \tag{1.17}$$

Where $\sigma_t$ is the stress that corresponds to the strain $\varepsilon$ , $\sigma_t^{'}$ represents the maximum stress, $\varepsilon_t^{'}$ represents the strain corresponding to the maximum stress $\sigma_t^{'}$, $\beta$ is a parameter depends on the shape of the stress-strain diagram.

An equivalent uniaxial stress-strain relations was provided by Chen [6] for biaxial and triaxial stress conditions of concrete. In the case of biaxial compression stress, Chen [6] suggest the following formula:

$$\sigma = \frac{E_0\varepsilon_{iu}}{1+(\frac{E_0}{E_s}-2)\frac{\varepsilon_{iu}}{\varepsilon_{ic}}+(\frac{\varepsilon_{iu}}{\varepsilon_{ic}})^2} \tag{1.18}$$

With:

| | |
|---|---|
| $E_0$ | Initial tangent modulus of elasticity |
| $E_S = \frac{\sigma_{ic}}{\varepsilon_{ic}}$ | Secant modulus at the maximum (peak) compressive stress |
| $\varepsilon_{ic}$ | Equivalent uniaxial strain corresponding to peak compressive principal stress |
| $\varepsilon_{iu}$ | Equivalent uniaxial strain |

For triaxial tension and compression case, Chen [6] suggest the next formula:

$$\sigma = \frac{E_0 \varepsilon_{iu}}{1 + (R + \frac{E_0}{E_s} - 2)\frac{\varepsilon_{iu}}{\varepsilon_{ic}} + (2R - 1)(\frac{\varepsilon_{iu}}{\varepsilon_{ic}})^2 + R(\frac{\varepsilon_{iu}}{\varepsilon_{ic}})^3} \tag{1.19}$$

With

$$R = \frac{E_0(\frac{\sigma_{ic}}{\sigma_{if}} - 1)}{E_s(\frac{\varepsilon_{ic}}{\varepsilon_{if}} - 1)^2} - \frac{\varepsilon_{ic}}{\varepsilon_{if}} \tag{1.20}$$

With $\sigma_{if}, \varepsilon_{if}$ represent the coordinates of several points on the descending branch of the stress-equivalent strain curve

## I.6 Damage Models

Damage models are often used to describe the concrete behavior in tension and compression. Continuum damage mechanics was suggested by Kachanov in the late ‘50s for creep related problems and was applied to the progressive failure of materials. The earlier form of this category of models was suggested by Dougill [53], [54], and define the plastic deformation through the flow rule and the stiffness degradation is modeled by fracturing theory. Later, a recent form of damage models was suggested using of a set of state variables computing the internal damage provided by an external load. Table 1.2 presents the damage parameters state in several constitutive models available in the literature. The main idea of these models is that the local damage in the concrete material can be represented by damage variables that are related to the tangential stiffness matrix. Various categories of damage models can be found in the literature such as elastic damage, plastic damage (Ju [55], Lubliner et al [26] Lee et al. [28]), and damage model using bounding surface concept (Voyiadjis [56]). In the late ‘80s, Krajcinovic [57] suggested using the damage mechanics to model accurately the strain-softening response of concrete. Similarly, Lubliner et al [26] suggested substituting the hardening variable in the overall form of classical plasticity by the plastic damage variable to describe the strain-softening response of concrete for both cases compression and tension.

**Table 1.2:** Representation of damage (Singh [58])

| Damage variable as | References |
|---|---|
| Scalar | Kachnov [59], Rabotov [60], Simo and Ju [61], [62], Ju [63] , Lemaitre [64]–[66], Chaboche [67], [68] |
| Vector | Krajcinovic and Foneska [69], Krajcinovic [70] |
| Second rank tensor | Kachanov, Dragon and Mroz, Cordebois and Sidoroff |
| Fourth order tensor | Chaboche [71], Ortiz [72] |
| Eight order tensor | Chaboche |
| Strain tensor | Bazant and Kim [11], Nicholson [73] |

### I.6.1 Damage Plastic Model

#### I.6.1.1 General description

One of the famous damage models is provided by Lubliner et al [26] under the name Plastic Damage Model for concrete. The main concept of this model is to substitute the hardening variable in the overall form of classical plasticity by the plastic damage variable which varies between two values (0 and 1), where the zero value represents the undamaged concrete and the value of one represents the totally damaged concrete with full loss of cohesion. The fundamental equations of this model are:

##### *I.6.1.1.a The Yield function*

$$F = \frac{1}{1-\alpha}(3\alpha p + \sqrt{3}J + \beta\langle\sigma_{\max}\rangle - \gamma\langle-\sigma_{\max}\rangle) - c \tag{1.21}$$

With $\alpha, \beta$ and $\gamma$ are dimensionless parameters given by:

$$\alpha = (\frac{f_{b0}}{f_{c0}} - 1) / (2\frac{f_{b0}}{f_{c0}} - 1) \tag{1.22}$$

$$\beta = R(1-\alpha) - (1+\alpha) \; with \; R = \frac{f_{c0}}{f_{t0}} \tag{1.23}$$

$$\gamma = 3(1 - r_{oct}^{\max}) / (2r_{oct}^{\max} - 1) \tag{1.24}$$

Here:

| | |
|---|---|
| $J$ | Deviatoric stress, |
| $p$ | Mean stress, |
| $\sigma_{\max}$ | The maximum principal effective stress, |
| $\frac{f_{b0}}{f_{c0}}$ | The ratio of biaxial and uniaxial compressive yield strengths. According to Wu and Faria [74], $f_{b0} / f_{c0}$ takes a value between 1.10 and 1.20. In ABAQUS [75], the default value is 1.16, |
| $f_{t0}$ | The initial uniaxial tensile yield stress, |
| $c$ | Cohesion, |
| $r_{oct}^{\max}$ | Constant takes a value of 0.65 according to Oller et al [30], |
| $< X >$ | Macaulay bracket and takes the form: $\langle \pm X \rangle = \frac{X \pm \lvert X \rvert}{2}$ |

***I.6.1.1.b The potential function:***

A non-associated potential plastic flow was suggested by Lubliner et al [26], where the potential function G takes the same form of the classical Mohr-Coulomb yield function with substituting the friction angle $\phi$ by the dilation angle $\psi$. It takes the following form:

$$G = p \sin \psi + J(\cos\theta - \frac{\sin\theta \sin\psi}{\sqrt{3}}) \tag{1.25}$$

Where:

| | |
|---|---|
| $J$ | Deviatoric stress, |
| $p$ | Mean stress, |
| $\psi$ | The dilation angle, |
| $\theta$ | Lode angle. |

The main weakness of the yield function and the potential function suggested by Lubliner et al [26] is their inability to handle the dynamic loading. Hence, a second form of this model was developed by Lee and Fenves [28] to address the dynamic loading. The following modifications were proposed:

**The Yield function**

- Substituting cohesion "$c$" by effective compressive cohesion stress $\overline{\sigma_c}$
- New formulas for $\beta$ parameter and zero value for $\gamma$ parameter. In our work $\gamma$ parameter will be considered, these parameters are given by:

$$\beta = \frac{\overline{\sigma_c}}{\overline{\sigma_t}}(1-\alpha) - (1+\alpha) \tag{1.26}$$

$$\gamma = \frac{3(1-K_c)}{2K_c - 1} \tag{1.27}$$

Where:

| | |
|---|---|
| $K_c$ | The ratio of second stress invariants on tensile and compressive meridians. For Mohr-Coulomb yield surface, $K_C$ takes a value of 0.7 [23]. |
| $\overline{\sigma_c}$ | The effective compressive cohesion stress |
| $\overline{\sigma_t}$ | The effective tensile cohesion stress |

As a result, the second form of the yield function suggested by Lee and Fenves [28] is written as:

$$F = \frac{1}{1-\alpha}\left(3\alpha\, p + \sqrt{3}\, J + \beta\langle\sigma_{\max}\rangle - \gamma\langle-\sigma_{\max}\rangle\right) - \overline{\sigma_c} \tag{1.28}$$

In order to extend the yield function suggested by Lee and Fenves [28] to handle the triaxial compression stress states, Zhang et al [76] proposed the next correlation for the yield function:

$$F = \frac{\sqrt{3J_2} + \left[\alpha H_1(\sigma_{\max}) + \delta H_0(-\sigma_{\max})\right] I_1 + \beta(k)\langle\sigma_{\max}\rangle - \gamma\langle-\sigma_{\max}\rangle}{1-\alpha H_1(\sigma_{\max}) - \delta H_0(-\sigma_{\max})} - \sigma_c(k) \tag{1.29}$$

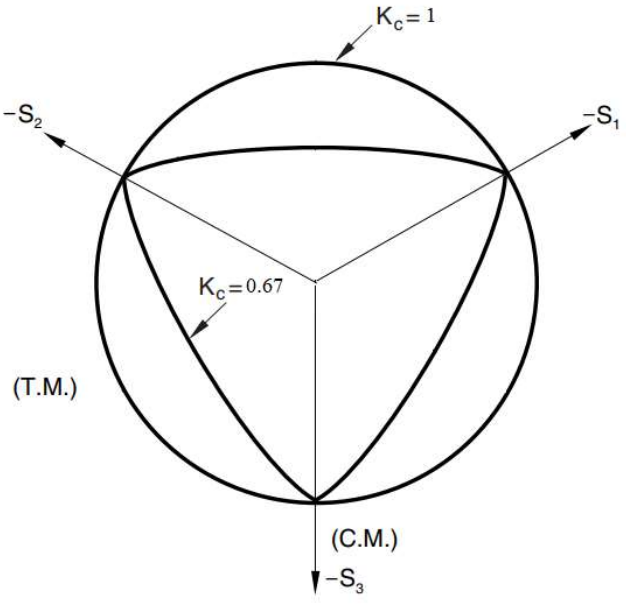


**Figure 1.1**: Yield surface in the deviatoric plane

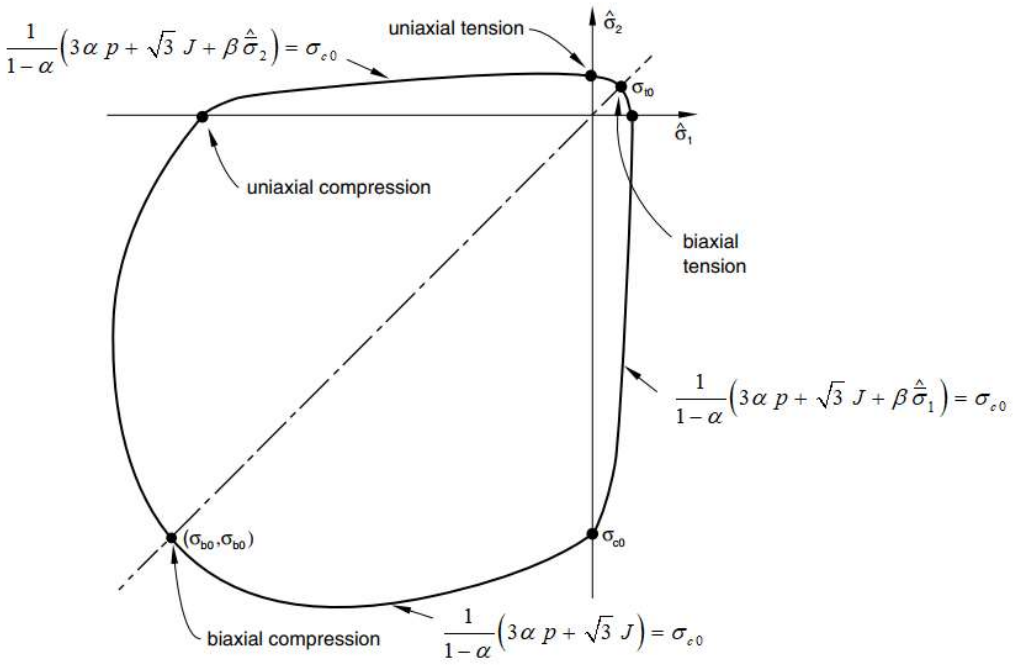


**Figure 1.2**: Yield surface in plane stress

With

$$H_\beta(x)=\begin{cases}0 & (x<0)\\ \beta & (x=0)\\ 1 & (x>0)\end{cases},\ \beta=\frac{c_c(k)}{c_t(k)}(1-\alpha)-(1+\alpha);\ \text{and}\ k=\begin{Bmatrix}k_t\\ k_c\end{Bmatrix}$$

Typical yield surfaces are shown in Figure 1.1 on the deviatoric plane and in Figure 1.2 for plane stress conditions.

**The potential function**

By using the Drucker-Prager hyperbolic function, a new formula for the potential function was developed by Lee and Fenves [28] given by:

$$G=\sqrt{(\varepsilon\sigma_{t0}\tan\psi)^2+3J^2}+p\tan\psi \qquad (1.30)$$

Here,

| | |
|---|---|
| $\sigma_{t0}$ | The uniaxial tensile stress at failure, |
| $\varepsilon$ | The flow potential eccentricity (0.1 in ABAQUS). |

According to the Model Code recommendations [33], the value of $\sigma_{t0}$ is related to the concrete grade:

**For concrete grade ≤ C50**

$$\sigma_{t0} = f_{tm} = 0.3016\, f_{ck}^{2/3} \tag{1.31}$$

**For concrete grade > C50**

$$\sigma_{t0} = f_{tm} = 2.12 \ln\left(1 + 0.1\left(f_{ck} + 8\right)\right) \tag{1.32}$$

With $f_{ck}$ is the characteristic value of concrete compressive strength,

Following ABAQUS user manual [75], the default value of the flow potential eccentricity is 0.1. This value indicates that the material has the same dilation angle over a wide range of confining pressure stress values. Values of the flow potential eccentricity that are greater than 0.1 provides more curvature to the flow potential, which means that the dilation angle increases more rapidly as the confining pressure decreases. Values of the flow potential eccentricity that are significantly less than the default value may lead to convergence problems according to ABAQUS user manual [75].

The estimation of the yield function "F" and the potential function "G" values is a step key in the finite element implementation of any non-linear constitutive model. For instance, Figure 1.3 shows the necessary steps adopted in the implementation of DPM for static loading.

To implement DPM, the following steps must be followed:

- Calculation of the yield function value in order to identify the material state which is elastic behavior if F < 0, plastic (or elastoplastic) behavior if F = 0 and impossible situation if F > 0. (The derivative of the yield function is used only to correct the stress state for F>0)
- Calculation of the potential function value and the derivative of the potential function to evaluate the plastic strain.

The required parameters to identify the yield function F and the potential function G are summarized in Table 1.3

### I.6.1.2 Estimation of the effective tensile and the effective compressive cohesion stress

In order to evaluate the value of the yield function, the parameters $\alpha$, $\beta$, and $\gamma$ must be identified using Eq(1.22) and Eq(1.27) for $\alpha$ and $\gamma$, respectively. From Eq(1.26), the estimation of $\beta$ is related to the values of each; the effective tensile and the effective compressive cohesion stresses that can be estimated by:

**Table 1.3:** Required parameters of DPM

| Parameter | Default value/Estimation methodology |
|---|---|
| $\sigma_{t0}$ | The uniaxial tensile stress at failure |
| $\varepsilon$ | The eccentricity |
| $\psi$ | The dilation angle |
| $K_c$ | The ratio of the second stress invariants on tensile and compressive meridians |
| $f_{b0} / f_{c0}$ | The ratio of biaxial compressive yield stress to uniaxial compressive yield stress. |
| $\sigma_c$ | The compressive stress |
| $\sigma_t$ | The tensile stress |
| $d_c$ | The compressive damage parameter |
| $d_t$ | The tensile damage parameter |

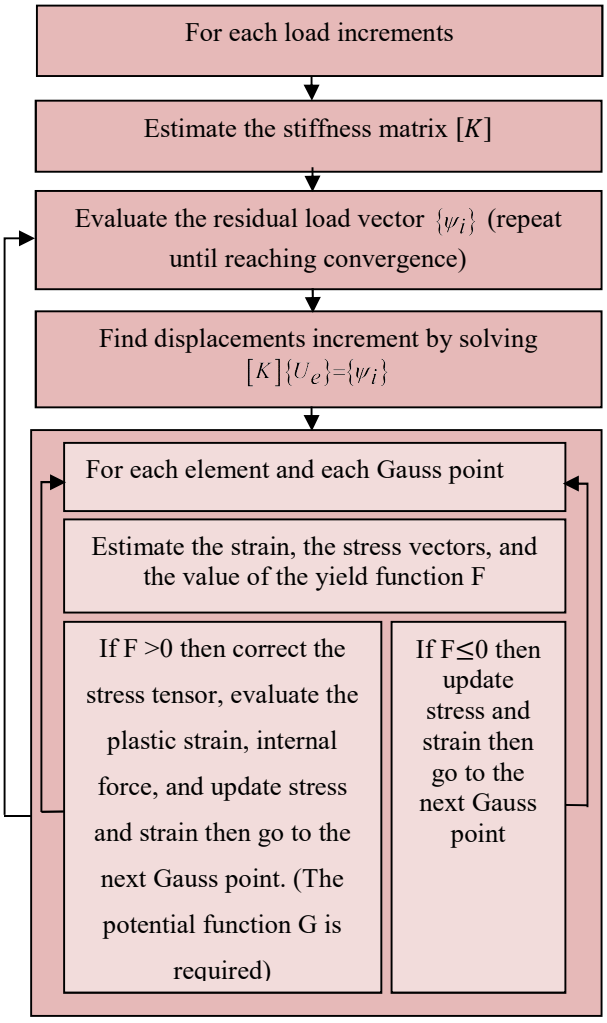


**Figure 1.3**: Implementation of DMP in "Concrete v2.0"

$$\begin{cases} \overline{\sigma_t} = \dfrac{\sigma_t}{1-d_t} \\ \overline{\sigma_c} = \dfrac{\sigma_c}{1-d_c} \end{cases} \tag{1.33}$$

Where

| | |
|---|---|
| $d_c$ | the compressive damage parameter |
| $d_t$ | the tensile damage parameter |
| $\sigma_c$ | the compressive stress |
| $\sigma_t$ | the tensile stress |

The compressive and tensile stresses can be evaluated from the stress-strain curves as shown in Figures 1.4 and 1.5.

To generate the stress-strain curves, two main alternatives can be used:

- User data: where the user supplied either the stress-inelastic strain data or the stress-strain data.
- Auto-estimation: where the stress-inelastic strain data or the stress-strain data are auto-evaluated basing on the characteristic value of concrete compressive strength.

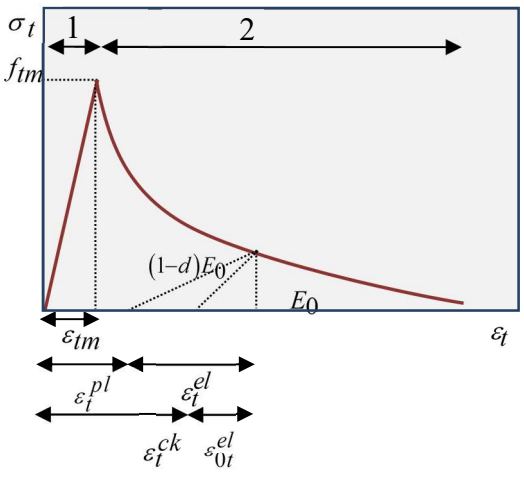


**Figure 1.4:** Response of concrete to uniaxial loading in tension

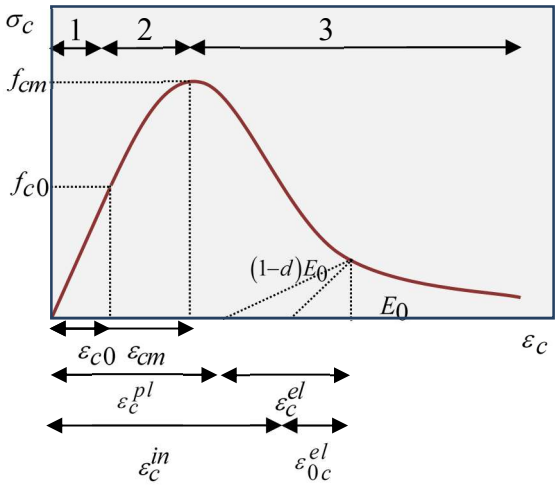


**Figure 1.5:** Response of concrete to uniaxial loading in compression

### *I.6.1.2.a User data*

In this approach, the stress-strain or the stress-inelastic strain curve is supplied by the user as a set of points. The strain-inelastic strain correlations are given by:

$$\begin{cases} \varepsilon_t = \varepsilon_t^{ck} + \dfrac{\sigma_t}{E_0} \\ \varepsilon_c = \varepsilon_c^{in} + \dfrac{\sigma_c}{E_0} \end{cases} \tag{1.34}$$

Where $E_0$ is the initial undamaged stiffness, $\varepsilon_t^{ck}$ is the tensile inelastic strain (the cracking strain), $\varepsilon_c^{in}$ is the compressive inelastic strain (crushing strain).

Also, the damage parameters in this approach can be estimated by:

- For the tensile damage parameter, Hafezolghorani et al [77] suggested a simple correlation takes the following form:

$$\begin{cases} d_t = 0 & if\ \varepsilon_t \le \varepsilon_{tm} \\ d_t = 1 - \dfrac{\sigma_t}{f_{tm}} & if\ \varepsilon_t > \varepsilon_{tm} \end{cases} \tag{1.35}$$

- For the compressive damage parameter, Hafezolghorani et al [77] and Yu et al [78] suggested the following correlation:

$$\begin{cases} d_c = 0 & if\ \varepsilon_c \le \varepsilon_{cm} \\ d_c = 1 - \dfrac{\sigma_c}{f_{cm}} & if\ \varepsilon_c > \varepsilon_{cm} \end{cases} \tag{1.36}$$

Finally, the damage parameter can be evaluated as:

$$d = 1 - (1 - s_t d_c)(1 - s_c d_t) \tag{1.37}$$

Where $s_c$ and $s_t$ are the stress state evaluated by:

$$\begin{cases} s_c = 1 - h_c\left(1 - r^*(\sigma_{11})\right) \\ s_t = 1 - h_t r^*(\sigma_{11}) \end{cases} \tag{1.38}$$

Where

$h_c$ and $h_t$ are weighting factors that varying between 0 and 1

$r^*(\sigma_{11})$ is the unit step. For uniaxial loading: $r^*(\sigma_{11}) = \begin{cases} 1 & if\ \sigma_{11} > 0 \\ 0 & if\ \sigma_{11} < 0 \end{cases}$ (1.39)

In the case of multiaxial loading conditions, the computing of the damage parameter is based on the same formula with replacing the unit step function $r^*(\sigma_{11})$ by the multiaxial stress weight factor $r(\sigma)$ which given by:

$$r(\sigma)=\frac{\sum_{i=1}^{3}\langle\sigma_i\rangle}{\sum_{i=1}^{3}|\sigma_i|};0\le r(\sigma)\le 1 \tag{1.40}$$

Another formula of the damage parameter provided by Demin and Fukang [79] takes the following form:

$$d=1-\sqrt{\frac{\sigma}{E_0\varepsilon}} \tag{1.41}$$

***I.6.1.2.b Auto-estimation***

Several research works were provided to auto evaluate the stress-strain diagrams and the stress-inelastic strain and the damage parameters evolution. Lubliner et al [26] suggested a simple approach for auto computing the stress- inelastic strain both in tension and in compression cases. According to Lubliner et al [26], the relations of stress- inelastic strain both in tension and in compression are given by:

$$\begin{cases}\sigma_c=f_{c0}\left[(1+a_c)e^{-b_c\varepsilon_c^{in}}-a_c e^{-2b_c\varepsilon_c^{in}}\right]\\ \sigma_t=f_{t0}\left[(1+a_t)e^{-b_t\varepsilon_t^{ck}}-a_t e^{-2b_t\varepsilon_t^{ck}}\right]\end{cases} \tag{1.42}$$

Where $f_{c0}$ and $f_{t0}$ are the compressive and tensile stresses that correspond respectively to zero crushing ( $\varepsilon_c^{in}=0$ ) and zero cracking ( $\varepsilon_t^{ck}=0$ ).

$a_c$, $a_t$, $b_c$, and $b_t$ are dimensionless coefficients evaluated from the correlation of tensile/compressive energies per unit of volume dissipated by damage along entire deterioration process (See Figures 1.6-1.7), which are given by:

$$\begin{cases}g_c=\int_0^{\infty}\sigma_c\ \mathrm{d}\varepsilon_c^{in}\\ g_t=\int_0^{\infty}\sigma_t\ \mathrm{d}\varepsilon_t^{ck}\end{cases} \tag{1.43}$$

By replacing the tension and compression stresses of Eq (1.42) in Eq (1.43), the following formula is obtained:

$$\begin{cases}g_c=\frac{f_{c0}}{b_c}\left(1+0.5\,a_c\right)\\ g_t=\frac{f_{t0}}{b_t}\left(1+0.5\,a_t\right)\end{cases} \tag{1.44}$$

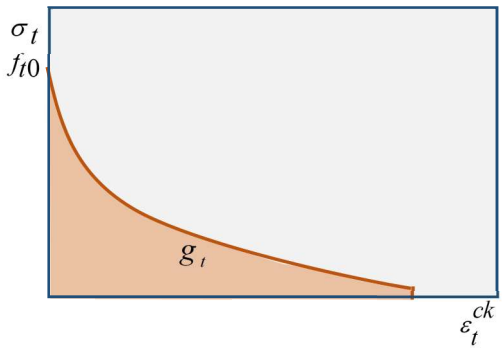


**Figure 1.6:** Parts of tension energy dissipated by damage

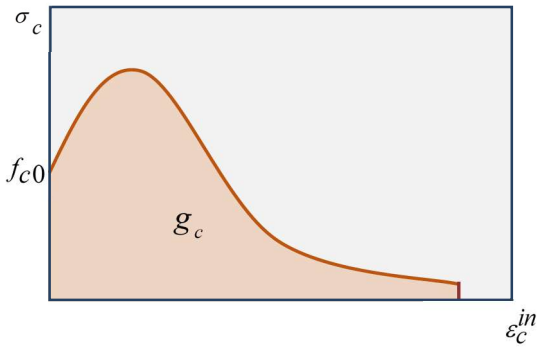


**Figure 1.7:** Parts of compressive energy dissipated by damage

Since $g_c = G_{ch} / L_{eq}$ and $g_t = G_F / L_{eq}$, the coefficients $b_c$ and $b_t$ take the following form:

$$b_c = \frac{f_{c0} L_{eq}}{G_{ch}} \left(1 + 0.5\ a_c\right) \tag{1.45}$$

$$b_t = \frac{f_{t0} L_{eq}}{G_F} \left(1 + 0.5\ a_t\right) \tag{1.46}$$

Where:

| | |
|---|---|
| $G_{ch}$ | The crushing energy per unit area. |
| $G_F$ | The fracture energy per unit area. |
| $L_{eq}$ | The mesh size (finite element characteristic length). For a brick element, the mesh size value is computed as the volume divided by the largest face area. |

According to Alfarah et al [23], by zeroing derivatives of $\sigma_c$ and $\sigma_t$ in Eq (1.42) with respect to the compressive and tensile inelastic strains, respectively, the maximum values $f_{cm}$ and $f_{tm}$ are obtained by:

$$f_{cm} = \frac{f_{c0}\left(1 + a_c\right)^2}{4a_c}; f_{tm} = \frac{f_{t0}\left(1 + a_t\right)^2}{4a_t} \tag{1.47}$$

As result, the coefficients $a_c$ and $a_t$ take the next forms:

$$a_c = 2\frac{f_{cm}}{f_{c0}} - 2\sqrt{\left(\frac{f_{cm}}{f_{c0}}\right)^2 - \frac{f_{cm}}{f_{c0}}} \tag{1.48}$$

$$a_t = 2\frac{f_{tm}}{f_{t0}} - 2\sqrt{\left(\frac{f_{tm}}{f_{t0}}\right)^2 - \frac{f_{tm}}{f_{t0}}} \tag{1.49}$$

To compute the damage parameters evolution, Lubliner et al [26] suggested the following correlation:

$$\begin{cases} d_c = \dfrac{1}{g_c}\displaystyle\int_0^{\varepsilon_c^{in}} \sigma_c \ \mathrm{d}\varepsilon_c^{in} \\ d_t = \dfrac{1}{g_t}\displaystyle\int_0^{\varepsilon_t^{ck}} \sigma_t \ \mathrm{d}\varepsilon_t^{ck} \end{cases} \tag{1.50}$$

A second approach for computing the stress-strain diagrams can be found in the literature suggested by Alfarah et al [23] where for the uniaxial compression loading, Alfarah et al [23] (same approach of Kratzig and Polling [29]) divided the stress-strain diagram into three parts as follow (Figure 1.5):

- First part which is linear (until $f_{c0}$):

$$\sigma_{c(1)} = E_0 \varepsilon_c \tag{1.51}$$

- Second part that between $f_{c0}$ and $f_{cm}$:

$$\sigma_{c(2)} = \frac{E_{ci}\dfrac{\varepsilon_c}{f_{cm}} - \left(\dfrac{\varepsilon_c}{\varepsilon_{cm}}\right)^2}{1 + \left(E_{ci}\dfrac{\varepsilon_{cm}}{f_{cm}} - 2\right)\dfrac{\varepsilon_c}{\varepsilon_{cm}}} f_{cm} \tag{1.52}$$

- Third part after $f_{cm}$:

$$\sigma_{c(3)} = \left(\frac{2 + \gamma_c f_{cm} \varepsilon_{cm}}{2 f_{cm}} - \gamma_c \varepsilon_c + \frac{\varepsilon_c^2 \gamma_c}{2\varepsilon_{cm}}\right)^{-1} \tag{1.53}$$

Where $E_{ci}$ is the modulus of deformation of concrete for zero stress.

$$\gamma_c = \frac{\pi^2 f_{cm} \varepsilon_{cm}}{2\left[\dfrac{G_{ch}}{L_{eq}} - 0.5 f_{cm}\left(\varepsilon_{cm}\left(1-b\right) + b\dfrac{f_{cm}}{E_0}\right)\right]^2}; \quad b = \frac{\varepsilon_c^{pl}}{\varepsilon_c^{in}} \tag{1.54}$$

| | |
|---|---|
| $G_{ch}$ | The crushing energy per unit area. |
| $L_{eq}$ | The mesh size (characteristic length). |
| $\varepsilon_c^{pl}$ | The compressive plastic strain. |
| $\varepsilon_c^{in}$ | The compressive inelastic strain |

For the tension case, Alfarah et al [23] suggested to split up the stress-strain curve into two part as shown in Figure 1.4, where the stresses can be evaluated by:

- First part until $f_{tm}$ :

$$\sigma_{t(1)} = E_0 \varepsilon_t \tag{1.55}$$

- Second part after $f_{tm}$ :

$$\sigma_{t(2)} = f_{tm}\left[\left[1+(c_1\frac{w}{w_c})^3\right]e^{-c_2\frac{w}{w_c}} - \frac{w}{w_c}\left(1+c_1^3\right)e^{-c_2}\right] \tag{1.56}$$

With:

$w$ is the crack opening, given by $w = \left(\varepsilon_t - \varepsilon_{tm}\right) L_{eq}$ (1.57)

$w_c$ is the crack opening at fracture, given by $w_c = 5.14\,\frac{G_F}{f_{tm}}$ (1.58)

$G_F$: The fracture energy by unit area. According to the Model code recommendations [33] $G_F$ takes the following form:

$$G_F = 0.073 f_{cm}^{0.18} \tag{1.59}$$

$c_1$ and $c_2$ are dimensionless coefficients. According to Hordijk [80] $c_1$ and $c_2$ take values equal to 3 and 6.93 respectively.

Alfarah et al [23] suggested a closed-form expression for the damage parameters given by:

$$\begin{cases} d_c = 1 - \frac{1}{2+a_c}\left[2\left(1+a_c\right)e^{-b_c\varepsilon_c^{in}} - a_c e^{-2b_c\varepsilon_c^{in}}\right] \\ d_t = 1 - \frac{1}{2+a_t}\left[2\left(1+a_t\right)e^{-b_t\varepsilon_t^{ck}} - a_t e^{-2b_t\varepsilon_t^{ck}}\right] \end{cases} \tag{1.60}$$

According to Alfarah et al [23], the coefficients $a_c$ and $a_t$ are evaluated by considering $f_{c0} = 0.4 f_{cm}$ and $f_{t0} = f_{tm}$ (using the Model code recommendations). As results, the values of $a_c$ and $a_t$ are 7.873 and 1, respectively. The coefficients $b_c$ and $b_t$ are estimated according to Eqs (1.45) ,(1.46)

In brief, Alfarah et al suggested a new algorithm in the aim to compute the stress-strain curves and damage parameters evolution that can be summarized in the following steps (All stress values are in MPa):

a- Enter each of; the concrete compressive strength $f_{ck}$, the mesh size $L_{eq}$, and the initial value of b is equal to 0.9.

b- Evaluate the compressive/tensile stress strength by: $f_{cm} = f_{ck} + 8$ and $f_{tm} = 0.3016\, f_{ck}^{2/3}$

c- State the strain at compressive stress strength as $\varepsilon_{cm} = 0.0022$

d- Evaluate the initial tangent modulus of concrete deformation $E_{ci} = 10000\, f_{cm}^{1/3}$ and the undamaged modulus of deformation $E_0 = E_{ci}\left(0.8 + 0.2\frac{f_{cm}}{88}\right)$.

e- Compute the crushing/fracture energy (N/mm) as $G_{ch} = \left(\frac{f_{cm}}{f_{tm}}\right)^2 G_F$ and $G_F$ from Eq(1.59)

f- Compute the critical crack opening from Eq(1.58)

g- Build the compressive stress-strain curve using Eqs (1.51), (1.52), and (1.53).

h- Build the tensile stress-strain curve using using Eqs (1.55),(1.56)

i- Compute the coefficients $b_c$ and $b_t$ by Eqs(1.45),(1.46) using the default values of $a_c$ and $a_t$ ( $a_c = 7.873$ and $a_t = 1$).

j- Compute the compressive/tensile damage parameters using Eq(1.60).

k- Calculate the compressive and tensile plastic strains:

$$\varepsilon_c^{pl} = \varepsilon_c^{in} - \frac{\sigma_c d_c}{E_0\left(1 - d_c\right)} \tag{1.61}$$

$$\varepsilon_t^{pl} = \varepsilon_t^{ck} - \frac{\sigma_t d_t}{E_0\left(1 - d_t\right)} \tag{1.62}$$

l- Calculate the average value of the ratio b using Eq(1.54) and compare it with the assumption in step A. Repeat until reaching convergence.

## I.7 Conclusion:

This chapter describes various categories of concrete constitutive models that can be used for modeling concrete behavior, each of linear and nonlinear elastic models, plasticity models, endochronic model, empirical models, and damage models were described in detail where the following points can be outlined:

- The linear elastic models provide accurate results for an elastic material subjected to small strains which is not the case of concrete material where the strain values can be important. Also for nonlinear elastic models, it can address the large strain but shows a major failure in the case of non-elastic materials.

- The category of plasticity models can perfectly address small and large strains of both elastic and plastic cases but on the other hand, two main inconveniences can be observed. The first is the non-ability of handling the concrete degradation (the damage evolution) which provides a large margin of error specifically after the peak point (on the stress-strain curve), the second is that the concrete behavior is not the same for tension and compression cases. As result, the use of plasticity models to describe the behavior of damaged concrete structures provides inaccurate outcomes.
- For complexity reasons, the endochronic model is no more supported by the scientific community. Furthermore, on the first hand, the use of this model for concrete needs several improvements specifically to evaluate the step time, but on the other hand, more suitable models can be found in the literature to describe the concrete behavior without the need to an additional improvement.
- The only suitable category of models that can be used to model damaged concrete structures is the Damage Plastic Models for several logical reasons, which are; first, the ability to address small and large strain, second, this category of models can be used to address the plasticity of concrete material, the third reason is that can be used to handle the concrete degradation, and so to evaluate the damage parameters, the fourth is the compatibility in the compression and the tension cases. As result, the development of "Concrete" will be based mainly on this type of models as demonstrated in the next chapter

# Chapter II : Finite element implementation of Damage Plastic Model

## II.1 Introduction:

The finite element implementation of the plastic damage model is a very complicated process due to the complexity of the related calculation such as the estimation of the plastic strain where the plastic multiplier must be identified. The two most common implementations of PDM were provided in the ’90s by Oller et al [30] and Lee and Fenves [31] where both forms of the PDM were coded. The first form of the PDM was implemented by Oller et al [30] using the yield function and the potential function provided by Lubliner et al [26]. The general procedure of the implementation process was delivered in the work of Oller et al [30] where the authors provided the estimation process of each; the plastic damage variables, the evolution of the internal variable of the cohesion, the internal friction angle, dilatancy angle, and the stiffness degradation. Unfortunately, the key steps for the development of our computer code are not provided in this work, especially the closed-form solutions of the plastic multiplier, the derivatives of the yield function and the derivative of the potential function. In the late ’90s, Lee and Fenves [31] suggested a return mapping algorithm in order to implement the second form of the PDM where the yield function and the potential function provided by the same authors in [28] were used. The full description of the implementation process of the second form of PDM was delivered in the work of Lee and Fenves [31] where they provided the estimation process of the plastic damage variables and the closed-form solution of the plastic multiplier in addition to the evaluation process of the tangent stiffness matrix. The algorithm provided in this work was used in the famous finite element code ABAQUS under the name Concrete Damage Plasticity Model (CDPM). In the other hand, the use of the CDPM requires multiple parameters namely; the stress-inelastic strain diagram for compression and tension cases, the damage parameters evolution for compression and tension cases, the ratio of the second stress invariants on tensile and compressive meridians, the eccentricity, the ratio of biaxial compressive yield stress to uniaxial compressive yield stress, and the dilation angle. The outcomes of ABAQUS strongly depend on the values of these parameters. Thus, all parameters must be calibrated with experimental tests. Furthermore, both the high complexity and sensitivity of the calibration process deeply diminish the CDPM efficiency. Few research works addressed this problem with the aim to reduce the number of parameters needed in the calibration process and identify their typical values. Szczecina and Winnicki [81] recommended the values of 0.0001 and 5 degrees as typical values for the viscosity parameter and the dilation angle, respectively. In a related study, the same authors [82] advised assessing the dilation angle and the fracture energy in compliance with the results obtained from Strut-and-Tie method and laboratory tests. Sümer and Aktaş [83] proposed a closed-form solution for evaluating the compressive damage parameter. As well as Demir et al. [84] examined the role of the viscosity

parameter in the numerical simulation of RC deep beams and concluded that 0.0005 is the typical value for the viscosity parameter. Likewise, Bhartiya et al [85] employed a new algorithm to evaluate the dilation angle. In a different manner, experimental results were exploited from Silva et al [86] to calibrate the dilation angle ψ, the eccentricity ε, the ratio $f_{b0}/f_{c0}$, the ratio $Kc$, and the viscosity parameter μ.

Important research efforts were employed to avoid the problems due to the complexity and the sensitivity of the calibration process of both the stress-inelastic strain and the damage parameters diagrams. Thus, Lubliner et al [26] proposed a closed-form solutions for both of; the stress-inelastic strain and the damage parameters. Similarly, closed-form solutions were developed by Alfarah et al [23] to compute the damage parameters and generate the stress-strain diagrams. Differently, Behnam et al [87] developed an analytical approach to evaluate the damage parameters evolution in terms of corresponding inelastic strains. Moreover, they elaborated an analytical approach to estimate the stress-strain diagram under compressive loading. Likewise, Yangjian et al [88] exploited an analytical approach to compute the stress-strain diagrams. Additionally, the authors created closed-form solutions to calculate the damage parameters evolution. While Bhartiya et al [85] used a new algorithm to evaluate the stress-strain diagrams.

In order to facilitate the use of the second form of the PDM, several enhancements in the finite element implementation of PDM were suggested in this chapter to minimize the number of the required parameters and to provide closed-form solutions for each of the plastic multiplier, the derivative of the yield function with respect to the stress tensor, the derivative of the yield function with respect to the compressive inelastic strain, and the derivative of the potential function with respect to the stress tensor. Furthermore, a full description of the finite element implementation of the PDM has been provided in this chapter including the used algorithms and the coding technique

## II.2 Oller's implementation of PDM

In 1990, Oller et al [30] published the first finite element implementation of PDM where the developed code handle the complex behavior of concrete using the classical plasticity theory provided an adequate yield function to simulate the tension and the compression responses of concrete. Thus, the cracking paths can be detected through the local damage effect, which is estimated by the evolution of the damage parameters in tension and compressions cases. This work described perfectly the estimation process of each; the plastic damage variables, the evolution of the internal variable of cohesion, and the internal friction angle. The work of Oller used the first form of the plastic damage model which can not address the dynamic loading.

### II.2.1 Fundamental equations:

The total strains increment can be divided into elastic and plastic parts, according to the next correlation:

$$\varepsilon = D_e^{-1}\sigma + \varepsilon^p = \varepsilon^e + \varepsilon^p \tag{2.1}$$

With: $D_e$ is the elastic constitutive matrix

In order to calculate the plastic strain increment, Oller suggested using the flow rule which can be defined for the general case of non-associated plasticity as:

$$\Delta\varepsilon^p = d\lambda \frac{\partial G}{\partial \sigma} = d\lambda g \tag{2.2}$$

Where $d\lambda$ is the plastic multiplier and $g$ is the plastic flow vector that presents the derivative of the potential function with respect to the stress tensor. Unfortunately, both the closed-form solutions of the plastic multiplier and the derivative of the potential function are not provided in the work of Oller

The incremental stress can be evaluated by:

$$\Delta\sigma = D^{ep}\Delta\varepsilon \tag{2.3}$$

With $D^{ep}$ represents the elastoplastic constitutive matrix given as:

$$D^{ep} = D_e - \frac{\left[D_e\left\{\frac{\partial G}{\partial \sigma}\right\}\right]\otimes\left[D_e\left\{\frac{\partial F}{\partial \sigma}\right\}\right]}{\left[\left\{\frac{\partial F}{\partial \sigma}\right\}D_e\left\{\frac{\partial G}{\partial \sigma}\right\}\right] + A} \tag{2.4}$$

Where A is the hardening/Softening parameter evaluated according to Potts and Zdravkovic [89] by the next formula

$$A = -\frac{1}{\lambda}\left\{\frac{\partial F}{\partial k}\right\}^T \{\Delta k\} \tag{2.5}$$

For perfect plasticity the derivative of the yield function $\left\{\frac{\partial F}{\partial k}\right\}$ takes a value equal to zero which provides zero value for the parameter A. Otherwise, the estimation of parameter A can be performed by:

$$A = -\frac{1}{\lambda}\left\{\frac{\partial F}{\partial k}\right\}^T \frac{\partial\{k\}}{\partial\{\varepsilon^p\}}\{\Delta\varepsilon^p\} \text{ where } \frac{\partial\{k\}}{\partial\{\varepsilon^p\}} = const \tag{2.6}$$

### II.2.2 Definition of the compressive and the tensile Plastic Damage Variables

For uniaxial tension and compression tests, Oller et al [30] suggested using Lubliner's [26] formulas in order to evaluate the compressive and the tensile damage variables. For the tensile case, the damage variable can be estimated by:

$$k_t = \frac{1}{g_t}\int_0^t \sigma_t d\varepsilon_t^{ck} \tag{2.7}$$

For the compressive case, the damage variable can be estimated by:

$$k_c = \frac{1}{g_c}\int_0^t \sigma_c d\varepsilon_c^{in} \tag{2.8}$$

With $g_t$ and $g_c$ are the specific plastic works, defined by the areas presented in Figures 1.6 and 1.7

For a multiaxial stress state, Oller et al [30] suggested the following formula to evaluate the damage variables (written in terms of principal stress and plastic strain):

$$\Delta k = h_k(\sigma, k, c)\Delta\varepsilon^p = \sum_{i=1}^{3}(h_{ki}\Delta\varepsilon_i^p) \tag{2.9}$$

With:

$$h_{ki} = \frac{1}{g_t^*}\langle\sigma_i\rangle + \frac{1}{g_c^*}\langle-\sigma_i\rangle; g_t^* = g_t\frac{\sum_{i=1}^{3}\langle\sigma_i\rangle}{\sigma_t}; g_c^* = g_c\frac{\sum_{i=1}^{3}\langle-\sigma_i\rangle}{\sigma_c}$$

Where $\sigma_c$ and $\sigma_t$ denote values obtained from uniaxial compression and tension tests, respectively.

### II.2.3 Estimation of the stiffness degradation:

Considering the stiffness degradation effects required revaluating the elastic secant constitutive matrix D following two internal variables: the elastic and the plastic degradation variables. Oller et al [30] suggested using the simplest assumption for elastic degradation based on a simple isotropic degradation variable $d^e$ where the secant constitutive matrix is modified by:

$$D(d^e) = (1 - d^e)D_0 \tag{2.10}$$

Where $D_0$ represents the initial stiffness matrix and $d^e = 1 - e^{-\phi w^{e,0}}$. With $w^{e,0}$ is the square of the undamaged energy norm of the strain and $\phi$ is a constant.

For plastic degradation, a simple one-parameter model has also been used in the work of Oller. The plastic degradation takes place only in the softening branch and the stiffness is then proportional to the cohesion. The secant constitutive matrix is thus given by:

$$D(d^p) = (1 - d^p) D(d^e) \tag{2.11}$$

Where $d^p = 1 - \frac{c}{c^{peak}}$

Where $c$ is the actual value of cohesion and $c^{peak}$ is the maximum cohesion value reached.

## II.3 Lee's Implementation of PDM

Based on the second form of the plastic damage model, Lee and Fenves al [31] suggested a new return-mapping algorithm for the finite element implementation of the PDM. The developed algorithm can be used for a broader range of plastic-damage models. A new numerically stress computation scheme was suggested for plane stress problems where rather than solving a multi-dimensional iteration problem (where each of the plastic multiplier and two principal stresses are typical independent unknowns), the dimensionless scalar variable is evaluated iteratively to overcome the complexity and numerical difficulties occurring in multi-dimensional iterations. Also, the limits of the scalar variable are derived to deliver a thinner range of iterations which reduce significantly the required number of iterations to achieve the convergence of the algorithm

### II.3.1 Fundamental equations:

**a- The correlations between the stress $\sigma$ and the effective stress $\overline{\sigma}$**

$$\sigma = (1 - d)\overline{\sigma} \tag{2.12}$$

$$\overline{\sigma} = D_0(\varepsilon - \varepsilon^p) = 2\mu(\varepsilon - \varepsilon^p) + \overline{\gamma}(\theta - \overline{\theta^p})I \tag{2.13}$$

Where $d$ represents the damage variable, $D_0$ represents the initial stiffness matrix, $\varepsilon$ is the total strain and $\varepsilon^p$ is the plastic strain which can be estimated according to the Drucker-Prager plastic flow rule as:

$$\Delta\varepsilon^{p} = d\lambda\frac{\partial G}{\partial\sigma} = d\lambda(\frac{\bar{s}}{\|\bar{s}\|} + \alpha_{p} I) \tag{2.14}$$

With $G = \sqrt{2J_{2}} + \alpha_{p} I_{1} = \|s\| + \alpha_{p} I_{1}$ (2.15)

$\mu$ and $\bar{\gamma}$ represent Lame's constants evaluated by $E/(2(1+\upsilon))$ and $E\upsilon/((1+\upsilon)(1-2\upsilon))$, respectively.

**b- Loading-unloading conditions**

The Loading-unloading conditions are expressed in terms of the yield function and the plastic multiplier by:

$$\begin{aligned} &F \le 0 \\ &\overline{d\lambda} \ge 0 \\ &\overline{d\lambda} F = 0 \end{aligned} \tag{2.16}$$

**c- The damage evolution law**

The damage evolution law is defined by a function of the damage variable and the principal effective stress

$$\Delta k = d\lambda H(\hat{\bar{\sigma}}, k) \tag{2.17}$$

### II.3.2 Stress computation

For a given value of the damage variable, the effective stress is calculated in such a way the plastic consistency condition must be satisfied (F=0), where a discrete version of the yield function is written as:

$$F(\bar{\sigma}_{n+1}, k_{n+1}) = \alpha(I_{1})_{n+1} + \sqrt{\frac{3}{2}}\left\|\bar{s}_{n+1}\right\| + \beta_{n+1} H((\hat{\bar{\sigma}}_{\max})_{n+1})(\hat{\bar{\sigma}}_{\max})_{n+1} - (1-\alpha)c^{c}_{n+1} \tag{2.18}$$

For three dimensional or plane strain cases, Eq (2.14) can be evaluated using the next correlation

$$\Delta\hat{\varepsilon}^{p} = d\lambda\left[\frac{1}{\left\|s^{tr}_{n+1}\right\|}\hat{\sigma}^{tr}_{n+1} + (\alpha_{p} - \frac{I^{tr}_{1}}{3\left\|s^{tr}_{n+1}\right\|})I\right] \tag{2.19}$$

With tr donates the trial stress tensor.

According to Lee and Fenves al [31], the return-mapping equation for the principal effective stress is given by:

$$\hat{\bar{\sigma}}_{n+1} = \hat{\bar{\sigma}}^{tr}_{n+1} - d\lambda\left[\frac{2\gamma\mu_0}{\left\|s^{tr}_{n+1}\right\|}\hat{\bar{\sigma}}^{tr}_{n+1} - \left(\frac{\mu_0 I^{tr}_t}{3\left\|s^{tr}_{n+1}\right\|} - 3K_0\alpha_p\right)I\right] \tag{2.20}$$

For three dimensional or plane strain cases , the following correlation to evaluate the plastic multiplier is suggested:

$$d\lambda = \frac{\alpha(I^{tr}_1)_{n+1} + \sqrt{\frac{2}{3}}\left\|s^{tr}_{n+1}\right\| - \bar{\beta}(\hat{\bar{\sigma}}^{tr}_1)_{n+1} - (1-\alpha)c^c_{n+1}}{9K_0\alpha_p\alpha + \sqrt{6}\mu_0 + \beta\left[2\left[\mu_0 / \left\|s^{tr}_{n+1}\right\|\right](\hat{\bar{\sigma}}^{tr}_1)_{n+1} - (I^{tr}_1)_{n+1} / 3\left\|s^{tr}_{n+1}\right\| + 3K_0\alpha_p\right]} \tag{2.21}$$

With $\bar{\beta} = \beta_{n+1}H((\hat{\bar{\sigma}}_{\max})_{n+1})$

### II.3.3 Plane stress formulation

According to Lee and Fenves [31], the plane stress version of the return-mapping equation is given by:

$$\hat{\bar{\sigma}}_{n+1} = \hat{\bar{\sigma}}^{tr}_{n+1} - d\lambda\left[2\mu_0\frac{\hat{\bar{\sigma}}_{n+1}}{\left\|\bar{s}_{n+1}\right\|} - \left(2\mu_0\tilde{\nu}\frac{I_1}{3\left\|\bar{s}_{n+1}\right\|} - 2\tilde{K}_0\alpha_p\right)I\right] \tag{2.22}$$

Where $\tilde{\nu} = (1-2\nu)/(1-\nu)$ and $\tilde{K}_0 = E_0/(2(1-\nu))$

In this case, the plastic multiplier can be evaluated according to the next formula:

$$d\lambda = \left(\frac{1-\xi}{\xi}\right)\frac{\xi_1\xi^2 + \xi_2\xi(2\tilde{\nu}-3)(1-\alpha)c^c_{n+1}}{6(2\alpha+\bar{\beta})(1-\xi)\tilde{K}_0\alpha_p - \sqrt{6}\mu_0(3-2\tilde{\nu}+2\tilde{\nu}\xi)} \tag{2.23}$$

With

$$\xi_1 = \bar{\beta}\tilde{\nu}(\hat{\bar{\sigma}}^{tr}_1 - \hat{\bar{\sigma}}^{tr}_2);\ \xi_2 = (3\alpha+\bar{\beta}\tilde{\nu})I^{tr}_1 + (3-2\tilde{\nu})\bar{\beta}\hat{\bar{\sigma}}^{tr}_1 - 2\tilde{\nu}(1-\alpha)c^c_{n+1} \tag{2.24}$$

Applying the condition $d\lambda \geq 0$ in Eq(2.23), the value of $\xi$ will be in the range:

$$\max(0, \min(\xi^a, \xi^b)) < \xi < \min(1, \max(\xi^a, \xi^b)) \tag{2.25}$$

Where

$$\xi^a = \frac{-\xi_2 + \sqrt{\xi_2^2 - 4\xi_1(2\tilde{\nu} - 3)(1-\alpha)c_{n+1}^c}}{2\xi_1}$$
$$\xi^b = 1 - \frac{3\sqrt{6}\mu_0}{6(2\alpha + \overline{\beta})\widetilde{K}_0\alpha_p + 2\sqrt{6}\tilde{\nu}\mu_0} \tag{2.26}$$

## II.4 Proposed finite element implementation of PDM

In order to minimize the number of the required parameter in the finite element implementation of the PDM, the auto-estimation strategy was used in our computer code for computing the stress-strain diagrams and the damage parameters evolutions. Also, default values were suggested for each of the following parameters; the dilation angle ψ, the eccentricity ε, the ratio $f_{b0}/f_{c0}$, and the ratio $Kc$. Two main approaches were described in chapter I in order to auto-estimate the stress-strain diagrams and the damage parameters evolutions using Eqs (1.45),(1.46),(1.54). Following these equations, the mesh size value affects the stress-strain diagrams and the damage parameters evolutions. Therefore, to examine the mesh size influence on the stress-strain curves and the damage parameters evolution suggested by Alfarah et al [23] and Lubliner et al [26] , both approaches were implemented in the computer code "Concrete v2.0.0" where these diagrams were generated based on multiple mesh size values. In addition, to overcome the mesh size sensitivity, a new numerical approach is suggested to estimate the stress-strain diagrams and the damage parameters evolution in accordance with the Model Code recommendations [33]. In this approach, the stress-inelastic strain was evaluated according to Lubliner et al [26] formulas, and the damage parameters evolution was estimated according to Alfarah et al [23] formulas. The main aim of this section is to provide a finite element class to model concrete behavior using PDM as a constitutive model and the Object-Oriented Programming paradigm (OOP) as a coding technique. In this section, the closed-form solutions of each; the plastic multiplier, the derivative of the yield function with respect of stresses, the derivative of the yield function with respect of inelastic strain, and the derivative of the potential function with respect of stresses are provided. Furthermore, a detailed description of the suggested class is delivered in this section, including the required, functions, subroutines, and fields, in addition to the used algorithms. The PDM class was developed as a key part of our computer software "Concrete" with the objective of modeling damaged concrete structures with a minimum number of required parameters.

### II.4.1 The proposed Algorithm to implement PDM

Several approaches were developed to solve nonlinear finite element problems like the constant stiffness method, the tangent stiffness method, the visco-plastic method, the Newton-Raphson method, the modified Newton-Raphson method. In our computer code "Concrete v2.0.0", the simplest approach (constant stiffness method) was used to implement the PDM. According to the used approach, the finite element implementation process of the PDM can be summarized as follows:

a- Apply the load increment $\{\Delta F_i\}$

b- Calculate the displacement increment. For static loading, the following system must be solved $[K]\{\Delta U_I\} = \{\Delta \psi_j\}$ (2.27)

For the first iteration (j=1) $\{\Delta \psi_j\} = \{\Delta F_i\}$

c- For each Gauss integration point evaluate the strain increment according to

$\{\Delta \varepsilon_i\} = [B]\{\Delta U_I\}$ (2.28)

d- Evaluate the stress increment according to $\{\Delta \sigma_i\} = [D]\{\Delta \varepsilon_i\}$ (2.29)

e- If $F(\sigma + \Delta\sigma_i, \varepsilon_c^{in} + \Delta\varepsilon_c^{in}) \le 0$ ; integration points stay elastic, go to step g:

f- The integration points reached the plastic surface. The stress must be corrected

g- Update the stress and the strain tensors

h- Calculate the residual load $\{\Delta \psi_{j+1}\} = \{\Delta \psi j\} - \{r\}$,

where: $r = \int_V [B]^T \{\Delta\sigma_i\} dV$ (2.30)

i- If $\frac{\|\Delta \psi_{j+1}\|}{\|\Delta F_i\|} > TOL$ goto step "b"

According to the described algorithm, two key steps must be achieved, the first one is the estimation of the value of the yield function for a given stress tensor and the second one is the correction of the stress tensor for points outside the yield surface. As described in Chapter I, the estimation of the yield function required multiple parameters that are illustrated in Table 1.3 and take the default values illustrated in Table 2.1. The most significant parameters are :

- The effective compressive cohesion stress
- The effective tensile cohesion stress

**Table 2.1:** Default values of DPM parameters

| Parameter | Default value/Estimation methodology |
|---|---|
| $\sigma_{t0}$ | Eqs.(1.31),(1.32) |
| $\varepsilon$ | 0.1 According to [75] |
| $\psi$ | $13^0$ According to [44] and $5^0$ According to [81] |
| $K_c$ | 0.67 According to [75] / 0.7 According to [23] |
| $f_{b0} / f_{c0}$ | 1.16 According to [75] |
| $\sigma_c$ | See the following sections |
| $\sigma_t$ | See the following sections |
| $d_c$ | See the following sections |
| $d_t$ | See the following sections |

The auto-estimation of the stresses and the damage parameters suggested by Alfarah [23] and Lubliner [26] based mainly on the values of four coefficients which are $a_c, b_c, a_t$ and $b_t$. These coefficients are related to the mesh size value according to Eqs (1.48),(1.45),(1.49), and (1.46), which affects the stresses and the damage parameters values as demonstrated in the next section.

### II.4.2 Influence of the mesh size on the damage parameters and the stress-strain diagrams

In order to examine the mesh size influence on the stress-strain curves and the damage parameters evolution suggested by Alfarah et al [23] and Lubliner et al [26], both approaches were implemented in our finite element code "Concrete v2.0.0" where the stress-strain curves and the damage parameters evolution were generated following multiple mesh size values. Figures: 2.1-2.6 were generated by "Concrete v2.0.0" to demonstrate the influence of the mesh size on:

- The compressive and the tensile stress-strain curves
- The damage parameters evolution for the compressive and the tensile cases.

For Lubliner's [26] approach, three values of the mesh size (200, 300, and 700 mm) were selected to examine their influence on the stress-inelastic strain curves. Figures 2.1 and 2.2 present the compressive and the tensile stress- inelastic strain curves generated according to Lubliner et al [26] approach for the same concrete compressive strength (25 MPa). For each mesh size value (200, 300, and 700 mm) the coefficients $a_c, b_c, a_t$ and $b_t$ were calculated based on Eqs (1.48),(1.45),(1.49), and (1.46), respectively. From these figures, it is observed that the mesh size value strongly affects the

stress-strain curve shape where the peak points of all compressive stress-strain curves approach the same compressive strength value while the inelastic strain value for each peak point is different.

For Alfarah [23] approach, four values of the mesh size were selected (50, 100, 200, and 400 mm) to demonstrate its influence on the stress-strain curves and the damage. Several observations can be outlined based on the figures 2.3-2.6:

- The third part of the compressive stress-strain curve and the second part of the tensile stress-strain curve are strongly influenced by the mesh size value.
- The compressive and tensile damage parameters evolutions are related to the mesh size value.
- The value of the effective compressive cohesion stress $\overline{\sigma_c}$ introduced in the yield function depends on the values of the compressive stress and the compressive damage parameter.
- The damage parameters are not compatible with the stress-strain diagrams in both cases compression and tension stresses. This conclusion is based on the fact that the formulas of the damage parameters used in Alfarah method and presented in Eq (1.60) are derived from the stress-strain formulas developed by Lubliner (Eq. (1.42)) while the used stress-strain diagrams are based on other formulas

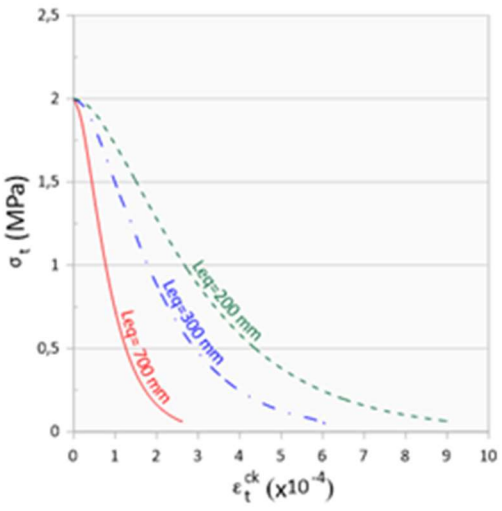


**Figure 2.1:** The influence of the mesh size on the tensile stress -inelastic strain, Lubliner approach

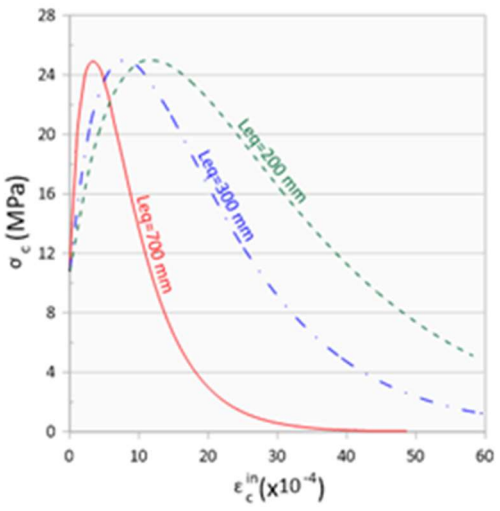


**Figure 2.2:** The influence of the mesh size on the compressive stress - inelastic strain, Lubliner approach

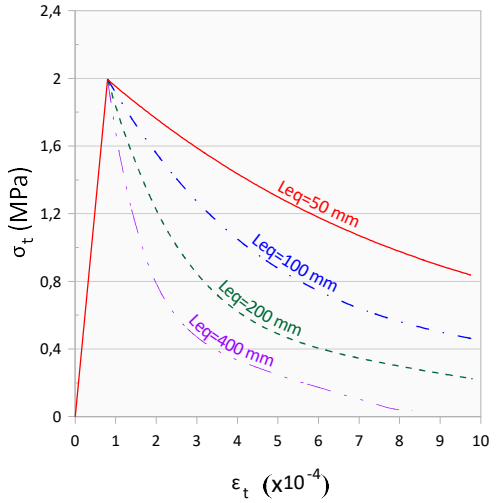


**Figure 2.3:** The influence of the mesh size on the tensile stress -strain, Alfarah approach

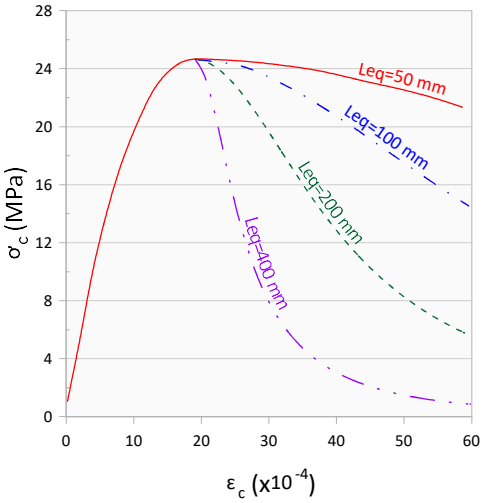


**Figure 2.4:** The influence of the mesh size on the compressive stress - strain, Alfarah approach

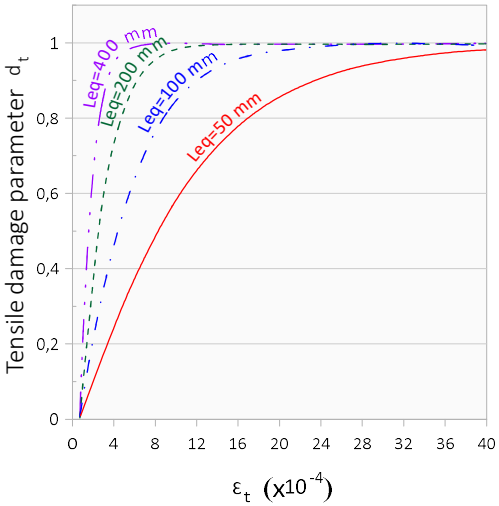


**Figure 2.5:** The influence of the mesh size on the tensile damage parameter

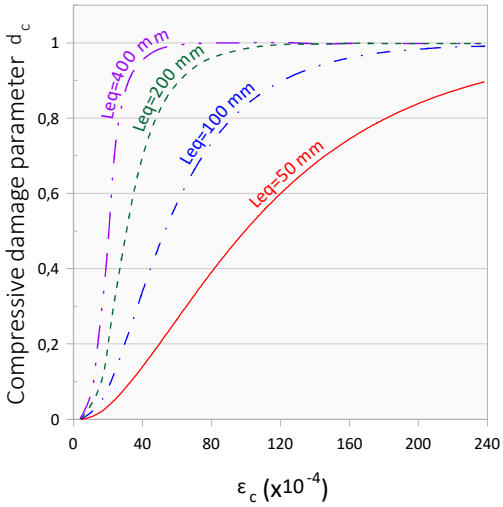


**Figure 2.6:** The influence of the mesh size on the compressive damage parameter

As result, the mesh size value will affect the outcomes of Alfarah approach (displacements, stress, and strain) for various logical reasons, which are:

- The value of β parameter delivered in the yield function depends on the effective compressive and tensile cohesion stress values. Therefore, it depends on both the values of the compressive and tensile stresses and the compressive and tensile damage parameters.

### II.4.3 The proposed approach for computing the damage parameters evolutions and the stress-strain diagrams

#### II.4.3.1 General description

The estimation of the stress-strain diagrams and the damage parameters evolution according to Eqs (1.42), (1.60), respectively depend mainly on the values of the coefficients $a_c, b_c, a_t$ and $b_t$ evaluated according to Eqs (1.48),(1.45), (1.49) and (1.46).As demonstrated previously, the values of these coefficients are deeply related to the mesh size value which affects the stresses and the damage parameters values. To overcome this issue and to evaluate the stress-inelastic strain curves according to Lubliner formulas and the damage parameters evolution in terms of inelastic strain according to Alfarah formulas, it is observed that by computing coefficients $a_c, b_c, a_t$ and $b_t$ according to Eqs (1.48),(1.45), (1.49) and (1.46) for the same concrete compressive strength and different mesh size values, the peak points of all compressive stress-strain curves approach the same compressive strength value while the strain value for each peak point is different (Figure 2.7), so it is suggested computing these coefficients in such a way the peak point will have the same strain and stress as delivered in the Model Code recommendations [33]. The main objective is to develop an algorithm that computes the previous coefficients that provide a peak point value equal to the peak point delivered in the model code recommendations. This algorithm computes the coefficients $a_c, b_c, a_t$ and $b_t$ according to the following steps:

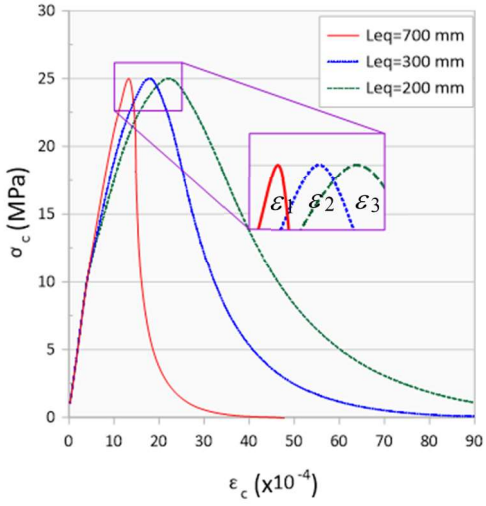


**Figure 2.7:** Mesh size effect on the Compressive stress vs compressive strain curve, case: $f_{cm}$=25*MPa*

- Evaluate $a_c$ and $a_t$ by substituting $f_{c0}$ with $0.4f_{cm}$ in Eq (1.48) and $f_{t0}$ with $f_{tm}$ in Eq (1.49). The values of $a_c$ and $a_t$ are 7.873 and 1, respectively
- Computing the coefficients $b_c$ and $b_t$ where the peak point has the same compressive strength and the same strain for compressive strength delivered in the Model Code recommendations [33]. Table 2.2 presents the strain values at peak stress for different concrete strength values according to the Model Code recommendations [33].

**Table 2.2:** Values of $\varepsilon_{c1}$ for different concrete strength (Model code [33])

| $f_{ck}$(MPa) | 12 | 16 | 20 | 25 | 30 | 35 | 40 | 45 | 50 | 55 | 60 | 70 | 80 | 90 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $f_{cm}$(MPa) | 20 | 24 | 28 | 33 | 38 | 43 | 48 | 53 | 58 | 63 | 68 | 78 | 88 | 98 |
| $\varepsilon_{c1}$(‰) | 1.8 | 1.9 | 2.0 | 2.1 | 2.2 | 2.25 | 2.3 | 2.4 | 2.45 | 2.5 | 2.6 | 2.7 | 2.8 | 2.8 |

### II.4.3.2 Stress-strain diagrams

In order to evaluate the stress-strain diagram under uniaxial compression loading, the diagram has been divided into two segments as shown in Figure 2.8 where:

- In the first (linear) segment (till $f_{c0}$), the compressive stress can be evaluated according to Hook's law
- In the second segment, the compressive stress and the compressive strain can be estimated by:

$$\begin{cases} \sigma_c = f_{c0}\left[\left(1+a_c\right)e^{-b_c\varepsilon_c^{in}} - a_c e^{-2b_c\varepsilon_c^{in}}\right] \\ \varepsilon_c = \varepsilon_c^{in} + \dfrac{\sigma_c}{E_0} \end{cases} \tag{2.31}$$

In the same manner, the estimation of the stress-strain diagram under uniaxial tensile loading based on the decomposition suggested in Figure 2.9 where:

- In the first (linear) segment (till $f_{tm}$), the tensile stress can be computed through Hook's law
- In the second segment, the tensile stress the tensile strain can be estimated by:

$$\begin{cases} \sigma_t = f_{t0}\left[\left(1+a_t\right)e^{-b_t\varepsilon_t^{ck}} - a_t e^{-2b_t\varepsilon_t^{ck}}\right] \\ \varepsilon_t = \varepsilon_t^{ck} + \dfrac{\sigma_t}{E_0} \end{cases} \tag{2.32}$$

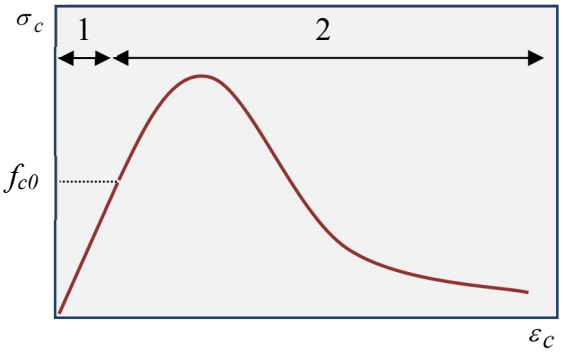


**Figure 2.8:** Response of concrete to uniaxial loading in compression

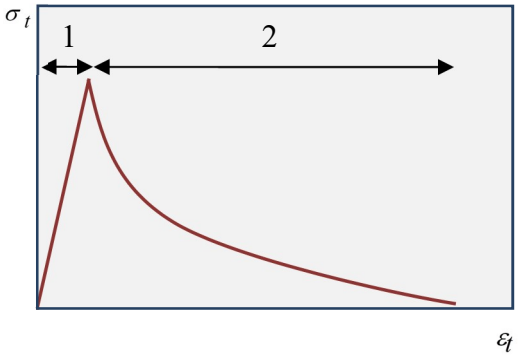


**Figure 2.9:** Response of concrete to uniaxial loading in tension

### II.4.3.3 Damage parameters evolution

In order to evaluate the compressive damage parameter evolution in terms of corresponding strain, the diagram has been divided into two parts, in which:

- In the first part, the compressive damage parameter value equal to zero
- In the second segment, the compressive damage parameter can be estimated according to:

$$\begin{cases} d_c = 1 - \dfrac{1}{2+a_c}\left[2\left(1+a_c\right)e^{-b_c\varepsilon_c^{in}} - a_c e^{-2b_c\varepsilon_c^{in}}\right] \\ \varepsilon_c = \varepsilon_c^{in} + \dfrac{\sigma_c}{E_0} \end{cases} \tag{2.33}$$

The same procedure can be used to compute the tensile damage parameter evolution in terms of the corresponding strain:

- In the first part, the tensile damage parameter value equal to zero
- In the second segment, the tensile damage parameter is evaluated by:

$$\begin{cases} d_t = 1 - \dfrac{1}{2+a_t}\left[2\left(1+a_t\right)e^{-b_t\varepsilon_t^{ck}} - a_t e^{-2b_t\varepsilon_t^{ck}}\right] \\ \varepsilon_t = \varepsilon_t^{ck} + \dfrac{\sigma_t}{E_0} \end{cases} \tag{2.34}$$

According to the Model Code recommendations [33], it is allowable to replace $f_{c0}$ by $0.4 f_{cm}$ and replace $f_{t0}$ by $f_{tm}$ in Eqs (1.48), (1.49), respectively, to obtain the values of $a_c$ and $a_t$, which are 7.873 and 1, respectively. The values of the coefficients $b_c$ and $b_t$, are estimated according to the algorithm described in the next section.

### II.4.3.4 The proposed algorithm for computing the coefficients $a_c$,$b_c$,$a_t$ and $b_t$

The main idea of the present algorithm is to determine the coefficient $b_c$ in such a way the peak point of the compressive stress-strain curve will have the same compressive strength and the same strain at compressive strength delivered in the Model Code recommendations [33]. The values of the strain at the peak stress delivered in the Model Code for different concrete strength values are summarized in table 2.2. In the model code recommendation, the strain values are estimated according to the following formula:

$$\varepsilon_{c1} = 0.5\, f_{cm}^{0.31} \le 2.8\ x10^{-3} \tag{2.35}$$

Where $f_{cm}$ is the compressive strength in MPa. According to the Model Code recommendations [33], the compressive strength can be estimated by:

$$f_{cm} = f_{ck} + 8 \tag{2.36}$$

In order to estimate coefficient $b_t$, we suggest computing the mesh size based on Eq (1.45) by:

$$L_{eq} = \frac{b_c G_{ch}}{f_{c0}\left(1+0.5\, a_c\right)} \tag{2.37}$$

By substituting Eq (2.37) in Eq (1.46), we can evaluate coefficient $b_t$ without the need to the mesh size value. The coefficient $b_t$ is given by:

$$b_t = b_c \frac{f_{t0}}{f_{c0}} \frac{G_{ch}}{G_F} \frac{1+0.5\, a_t}{1+0.5\, a_c} \tag{2.38}$$

**Table 2.3:** Values of coefficients $a_c$,$a_t$,$b_c$ and $b_t$ for different concrete strength –Part1

| $f_{ck}$(MPa) | 12 | 16 | 20 | 25 | 30 | 35 | 40 |
|---|---|---|---|---|---|---|---|
| $a_c$ | 7.873 | 7.873 | 7.873 | 7.873 | 7.873 | 7.873 | 7.873 |
| $b_c$ | 637.077 | 636.468 | 638.065 | 641.894 | 646.876 | 652.439 | 658.218 |
| $a_t$ | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| $b_t$ | 6122.778 | 6059.292 | 6107.316 | 6240.193 | 6412.655 | 6604.052 | 6803.804 |

**Table 2.4:** Values of coefficients $a_c$,$a_t$,$b_c$ and $b_t$ for different concrete strength –Part2

| $f_{ck}$(MPa) | 45 | 50 | 55 | 60 | 70 | 80 | 90 |
|---|---|---|---|---|---|---|---|
| $a_c$ | 7.873 | 7.873 | 7.873 | 7.873 | 7.873 | 7.873 | 7.873 |
| $b_c$ | 663.972 | 669.533 | 674.783 | 679.639 | 687.945 | 698.146 | 794.836 |
| $a_t$ | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| $b_t$ | 7005.913 | 7206.661 | 7403.612 | 7595.106 | 7957.263 | 8334.567 | 9769.134 |

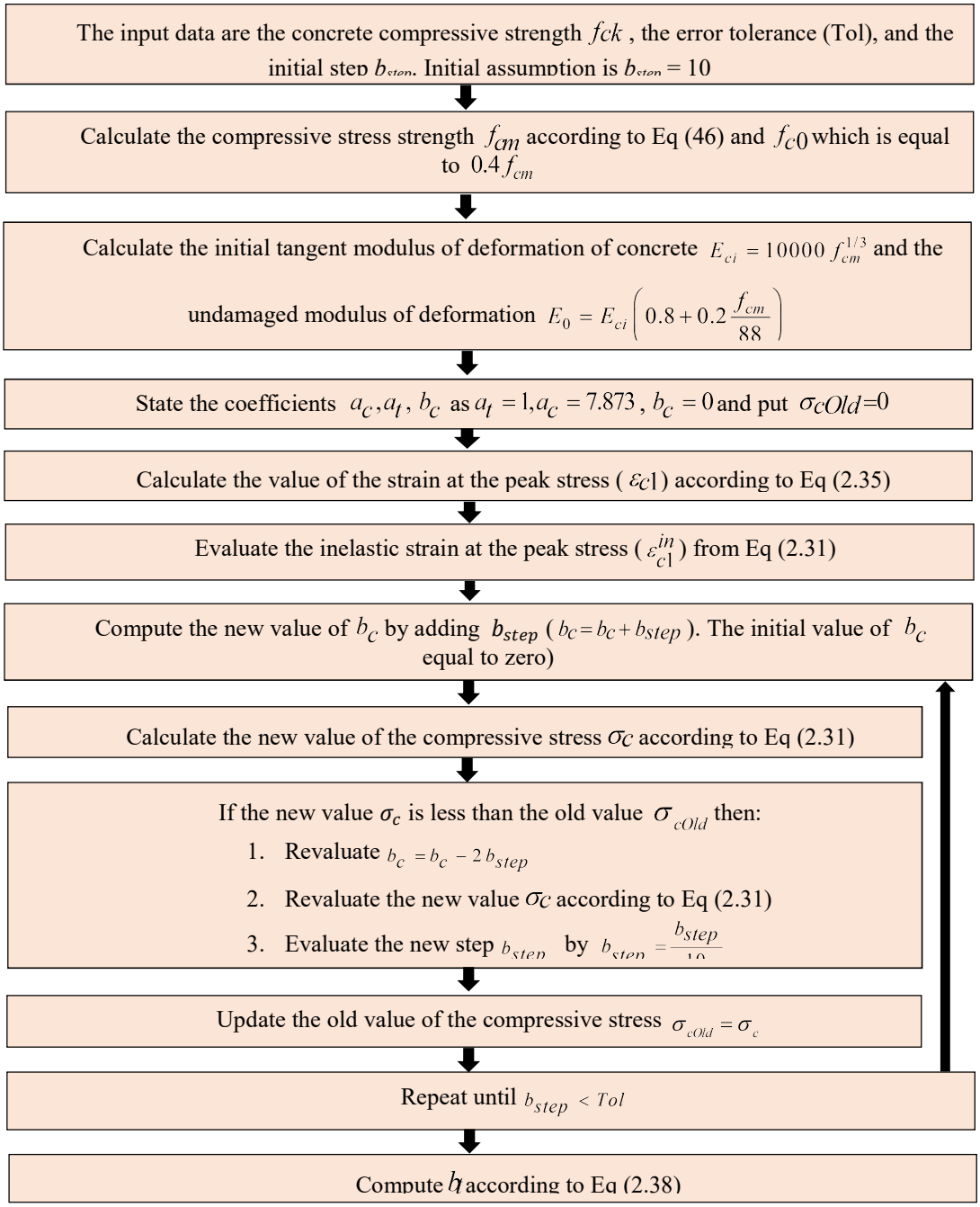


**Figure 2.10:** Proposed algorithm for evaluating $a_c, a_t, b_c$ , and $b_t$

The proposed algorithm illustrated in Figure 2.10 (All stress values are in MPa) was implemented in our computer code "Concrete v2.0.0" in order to examine the generated stress-strain diagrams and the damage parameters evolution. The values of coefficients $a_c, a_t, b_c$ and $b_t$ are summarized in tables 2.3 and 2.4 for different concrete strengths.

### II.4.4 The incremental stress and strain calculation:

In the plasticity theory, the total strains increment $\{\Delta\varepsilon\}$ can be divided into elastic $\{\Delta\varepsilon^e\}$ and plastic $\{\Delta\varepsilon^p\}$ parts, as follows:

$$\{\Delta\varepsilon\} = \left\{\Delta\varepsilon^{e}\right\} + \left\{\Delta\varepsilon^{p}\right\} \tag{2.39}$$

Where the plastic component can be evaluated according to the flow rule as:

$$\left\{\Delta\varepsilon^{p}\right\} = d\lambda\left[(1-\omega)(\frac{\partial G}{\partial\sigma})^{i-1} + \omega(\frac{\partial G}{\partial\sigma})^{i}\right] \tag{2.40}$$

Where $d\lambda$ is the plastic multiplier and $\omega$ is a parameter that depends on the type of time integration used.

For ω = 0 the integration is called explicit in which the derivative of the potential function is evaluated at point A (figure 2.11). In this case, the plastic strain can be estimated using the following schemes:

- Modified Euler
- Single-step modified Euler
- Dormand-Prince
- Runge-Kutta integration

For ω = 1 the integration is called implicit which means that the derivative of the potential function must be computed at point B (figure 2.11 ) where the following schemes can be used to estimate the plastic strain:

- Single-Step Backward Euler Scheme
- Backward Euler Return Scheme
- Return algorithm proposed by Ortiz & Simo (1986)
- Return algorithm proposed by Borja & Lee (1990)

In our computer code, the single-step Backward Euler scheme was selected to evaluate the plastic strain. In this algorithm, the plastic strain is evaluated at point B (figure 2.11) by the following correlation:

$$\left\{\Delta\varepsilon^{p}\right\} = d\lambda\frac{\partial G}{\partial\sigma} \tag{2.41}$$

The incremental stress $\{\Delta\sigma\}$ can be estimated from the incremental elastic strain $\left\{\Delta\varepsilon^{e}\right\}$ and the elastic constitutive matrix [*D*] according to the next formula:

$$\{\Delta\sigma\} = [D]\left\{\Delta\varepsilon^{e}\right\} \tag{2.42}$$

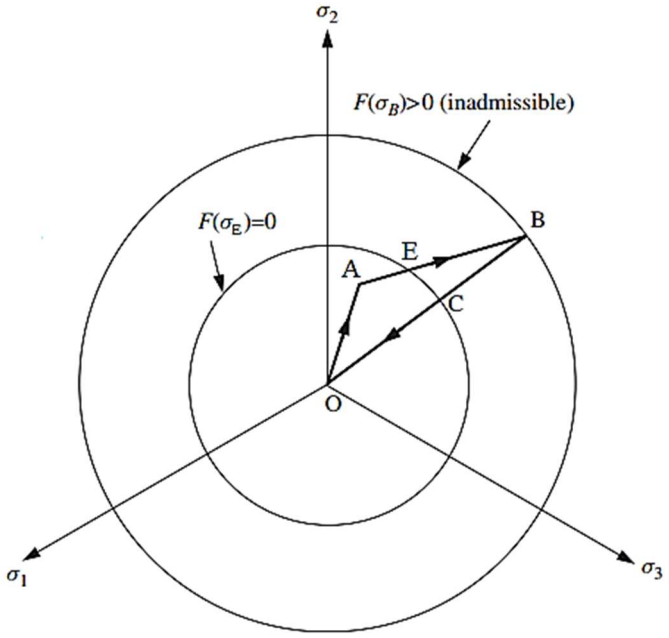


**Figure 2.11:** Stress correction

By using Eqs (2.39), (2.41), and (2.42), the incremental stress $\{\Delta\sigma\}$ takes the following form:

$$\{\Delta\sigma\} = [D]\{\Delta\varepsilon\} - d\lambda[D]\frac{\partial G}{\partial \sigma} \tag{2.43}$$

In plastic condition, the stress state in each integration point must always remain on the edge of the elastic domain (F=0). The Kuhn Tucker plasticity consistency condition is given by [90]:

$$F \le 0, \overline{d\lambda} \ge 0, \overline{d\lambda} F = 0, \overline{d\lambda} dF = 0 \tag{2.44}$$

By differentiating Eq (1.28) with respect to time, and using the chain rule of differentiation, the consistency condition becomes as follows [35]:

$$dF = \frac{\partial F}{\partial \sigma}\Delta\sigma + \frac{\partial F}{\partial \varepsilon_c^{in}}\Delta\varepsilon_c^{in} = 0 \tag{2.45}$$

From Eqs (2.43) and (2.45), the plastic multiplier $d\lambda$ can be evaluated according to the next correlation:

$$d\lambda = \frac{\left\{\frac{\partial F}{\partial \sigma}\right\}^T [D]\{\Delta\varepsilon\} + \frac{\partial F}{\partial \varepsilon_c^{in}}\Delta\varepsilon_c^{in}}{\left\{\frac{\partial F}{\partial \sigma}\right\}^T [D]\left\{\frac{\partial G}{\partial \sigma}\right\}} \tag{2.46}$$

The finite element implementation of the PDM model required the identification of the plastic multiplier which needs the estimation of each:

a- The derivative of the yield function with respect of stresses,
b- The derivative of the yield function with respect of the inelastic compression strain,
c- The derivative of the potential function with respect of stresses.

### II.4.4.1 The derivatives of the yield function:

In order to evaluate the plastic multiplier $d\lambda$, each of the derivative of the yield function with respect of stresses and the derivative of the yield function with respect to the compressive inelastic strain should be calculated. To evaluate the derivative of the yield function with respect of stresses, the Chain rule was used where the derivative becomes:

$$\frac{\partial F}{\partial \sigma} = \frac{\partial F}{\partial p}\frac{\partial p}{\partial \sigma} + \frac{\partial F}{\partial J}\frac{\partial J}{\partial \sigma} \tag{2.47}$$

Here:

$p$: Mean stress,

$J$: Deviatoric stress,

To evaluate the derivative of the yield function with respect to stress tensor, each term in Eq(2.47) must be determined. $\frac{\partial p}{\partial \sigma}$ and $\frac{\partial J}{\partial \sigma}$ are model-independent evaluated by Potts and Zdravkovic [89] as follows:

$$\frac{\partial p}{\partial \sigma} = \frac{1}{3}\{1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0\}^T \tag{2.48}$$

$$\frac{\partial J}{\partial \sigma} = \frac{1}{2J}\left\{\sigma_x - p \quad \sigma_y - p \quad \sigma_z - p \quad 2\tau_{xy} \quad 2\tau_{xz} \quad 2\tau_{yz}\right\}^T \tag{2.49}$$

$\frac{\partial F}{\partial p}$ and $\frac{\partial F}{\partial J}$ can be evaluated according to:

$$\frac{\partial F}{\partial p} = \frac{3\alpha}{1-\alpha} \tag{2.50}$$

$$\frac{\partial F}{\partial J} = \frac{\sqrt{3}}{1-\alpha} \tag{2.51}$$

The derivative of the yield function with respect to the compressive inelastic strain can be determined by:

$$\frac{\partial F}{\partial \varepsilon_c^{in}} = \left(\frac{\left\langle \sigma_{max} \right\rangle}{\overline{\sigma_t}} - 1\right) \frac{\partial \overline{\sigma_c}}{\partial \varepsilon_c^{in}} \tag{2.52}$$

With

$$\frac{\partial \overline{\sigma_c}}{\partial \varepsilon_c^{in}} = \frac{\partial (\sigma_c / (1 - d_c))}{\partial \varepsilon_c^{in}} = f_{c0} \frac{a_c (a_c + 1)(a_c + 2) b_c e^{b_c \varepsilon_c^{in}}}{((2a_c + 2) e^{b_c \varepsilon_c^{in}} - a_c)^2} \tag{2.53}$$

#### II.4.4.2 The derivative of the potential function:

The second step in the estimation of the plastic multiplier $d\lambda$ is to determine the derivative of the potential function where the Chain rule was used. The derivative of the potential function with respect to the stress tensor becomes:

$$\frac{\partial G}{\partial \sigma} = \frac{\partial G}{\partial p} \frac{\partial p}{\partial \sigma} + \frac{\partial G}{\partial J} \frac{\partial J}{\partial \sigma} \tag{2.54}$$

As we have seen previously $\frac{\partial p}{\partial \sigma}$ and $\frac{\partial J}{\partial \sigma}$ can be determined according to Eqs(2.48),(2.49) respectively. The estimation of $\frac{\partial G}{\partial p}$ and $\frac{\partial G}{\partial J}$ is based on:

$$\frac{\partial G}{\partial p} = \tan\psi \tag{2.55}$$

$$\frac{\partial G}{\partial J} = \frac{3J}{\sqrt{(\varepsilon \sigma_{t0} \tan\psi)^2 + 3J^2}} \tag{2.56}$$

## II.5 Conclusion:

In this chapter, the full procedure of the finite element implementation of the PDM was delivered, in addition to a new methodology to minimize the number of the required parameters. Only the compressive concrete strength is required to model a concrete sample for both cases compression and tension. Each of the stress-inelastic strain diagrams and the damage parameters evolution for compression and tension cases, the ratio of the second stress invariants on tensile and compressive meridians, the eccentricity, the ratio of biaxial compressive yield stress to uniaxial compressive yield

stress, and the dilation angle were auto-estimated and default values were suggested in this chapter. Also, the mesh size influence was eliminated in the proposed approach for computing the damage parameters evolutions and the stress-strain diagrams.

The stress-strain curves and the damage parameters evolution in tension and compression states were calculated in accordance with the Model Code recommendations. The main advantage of this approach is that the use of the Damage Plastic Model is no longer related to the complicated calibration process of the stress-strain and the damage parameters evolution with experimental tests. In fact, the only parameter needed in the developed approach is the concrete compressive strength value.

The plastic strain estimation process has been described in this chapter, where the closed-form of the plastic multiplier, the derivative of the yield function with respect of stresses, the derivative of the yield function with respect of the inelastic compression strain; and the derivative of the potential function with respect of stresses were provided

# Chapter III : Description of the finite element computer code “Concrete v2.0.0”

## III.1 Introduction

In order to achieve the aim of the present work, a new finite element computer code under the name “Concrete” was built to model damaged concrete structures with the minimum number of required parameters. For the first version of “Concrete 1.0.0”, the visual studio 2019 was used to build the App using visual basic.Net (Figure 3.1), and for the second version “Concrete v2.0.0”, the code was migrated to the last version of visual studio (visual studio 2022) in order to benefit from their recent features. Also, several third-party libraries were employed to accomplish multiple tasks such as the drawing process, meshing process, the user graphical interface, and performing the math calculations. The Triangle.Net library was used in our computer code to generate 2D mesh, this well-known library was developed by Woltering [91] as a port of the Triangle program made by Shewchuk under MIT licensing. In the same manner, a second library under the name Open Toolkit “OpenTK” [92] was used in “Concrete v2.0.0” to handle the drawing process with a high level of efficiency, this library gives the developer the ability to access the graphics card and perform the drawing with a high level of speed and quality. The Open Toolkit is distributed under the permissive MIT/X11. Also, Ribbon WinForms [93] is the third library that was used in our code in order to add a Ribbon to our computer code and so, improve the graphical user interface of “Concrete v2.0.0”. The last library that was used in “Concrete v2.0.0” is Math.Net Numerics [94] which is covered under the terms of the MIT license. This library was used with the aim to help the developer to carry out the math calculations.

To improve the quality of “Concrete 2.0.0”, the Object-Oriented Programming (OOP) paradigm was selected to use as a coding technique where instead to decompose the main program into a collection of variables, data structures, and subroutines, the program was decomposed into “Objects” that expose behavior and data using interfaces. This strategy helps us to improve the code quality where we can benefit from modern technology to getting a modern design, increase programs performance, debugging and error handling, and update managing.

In the computer code “Concrete v2.0.0”, the second form of the plastic damage model was selected as constitutive law to simulate the real behavior of concrete material. The implementation process of the PDM in “Concrete v2.0.0” was described in chapter II, where the only required parameter to use PDM is the compressive concrete strength. The stress-strain diagrams and the damage parameters evolutions were auto-calculated through our new approach. Moreover, typical values for the ratio of biaxial and uniaxial compressive yield strengths, The ratio of second stress

invariants on tensile and compressive meridians, the flow potential eccentricity, and the dilation angle were selected from the literature.

“Concrete v2.0.0” was developed under visual studio with vb.net based on OOP and the finite element fundamentals, it was developed to model cubical and cylindrical concrete structures. The eight-node brick element (C3D8) was used in our code to discretize and analyze the continuum. The current version enables the users to analyze structures with linear material properties and nonlinear material properties using PDM constitutive model.

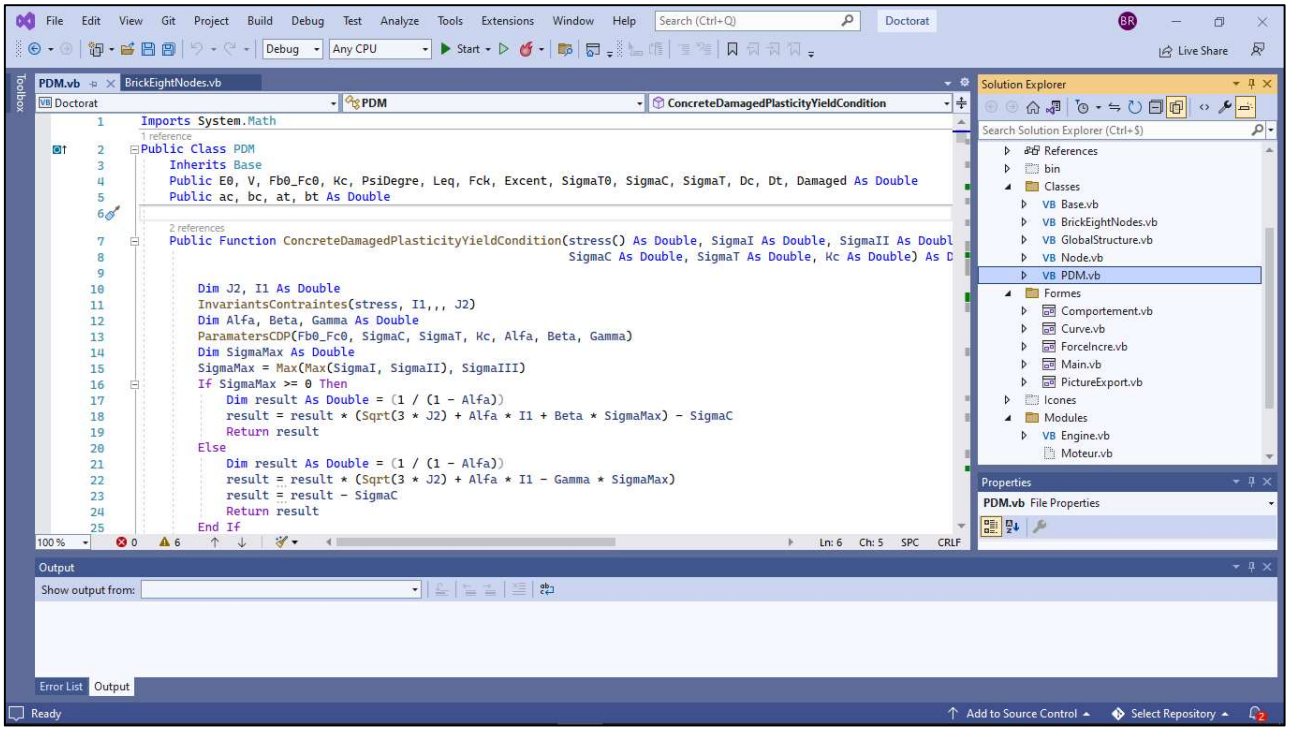


**Figure 3.1:** Code source of Concrete v2.0.0. Screenshot

## III.2 Object-Oriented Programming Paradigm

In the literature, the coding of linear and nonlinear finite element methods is mainly based on the Procedure Oriented Programming (POP) paradigm that appeared in the late 1950s with the programming languages ALGOL 58 and ALGOL 60, where the main idea is to divide the main programs into smaller self-contained program segments such as block structures and subroutines. With this technique, the FE application made great strides in the 1970s and 1980s, when a large amount of software was developed. Unfortunately, the POP demonstrated a huge failing in terms of software design, management of recent technology, code updating, and maintenance. With the development of a new programming paradigm called Object-Oriented Programming (OOP), a large

number of trade codes were raised such as PLAXIS, ABAQUS, ROBOBAT, SAP. In fact, the OOP was initiated for the first time in the late 1960s by the Norwegian developers O. J. Dahl and K. Nygaard [95] who developed a new programming language called SIMULA (SIMUlation LAnguage), based on the assumption that the main program should be modeled around objects rather than procedures. In fact, they invented the idea of "classes" in order to develop objects sharing similar characteristics. Therefore, these objects could communicate and make requests to each other. The OOP paradigm is based on four concepts:

- Encapsulation: grouping the data and the methods working with it within one unit;
- Abstraction: Objects only reveal internal mechanisms relevant to the use of other objects, masking any unnecessary implementation code;
- Inheritance: is a mechanism whereby a class acquires the property (fields and methods) of another class;
- Polymorphism: the object can take many forms. The most common use of this concept in OOP occurs when a parent class reference is used to refer to a child class object.

The main difference between POP and OOP is that the first approach aims to decompose the program into a collection of variables, data structures, and subroutines, while the second approach consists of decomposing the program into “Objects” that expose behavior and data using interfaces. Therefore, the most important difference is that POP uses procedures to operate on data structures, while OOP groups the two together. As a result, using OOP improves the code organization and increases the maintainability and reusability of the source code.

In the FE coding, several research projects have already suggested using the OOP paradigm, in which various programming languages were employed such as C++, Java, Matlab. In the early ‘90s, Forde et al [96] explained the possible solutions that object-oriented programs offer in the FE problems. Similarly, Mackie [97] described an object-oriented implementation of the finite element method and demonstrated its advantages. Likewise, Dubois-Pelerin et al [98] used the prototyping language Smalltalk for the OOP implementation of the finite element method. Later Kumar [99] suggested the implementation of OOP to the FEM for engineering analysis using C++. Also, Phillipe D.Alves et al [100] exploited the OOP to the generalized finite element method. Also, Benjamin et al [101] used the OOP to provide an FE toolbox within the Matlab environment.

In this work, the FE implementation of the PDM was performed through OOP paradigm. The architecture of “Concrete v2.0.0” shown in Figure 3.2 illustrates four classes used in its development which are the “Global Structure” class, “Brick Eight Nodes” class, “Node” class, and the “PDM” Class. In fact, the main class “Global Structure” holds several fields, subroutines, and functions in

order to model the structure such as the meshing process, computing the global stiffness matrix, load vector, and solving the system...etc. Also, the Global Structure contains a list of elements, each element is based on the Brick Eight Nodes class to evaluate the element stiffness matrix, the stress vector, and the strain vector based on:

- The position of each node, which can be found in the list of nodes where each node represents a node object.
- The material behavior (found in the PDM Class).

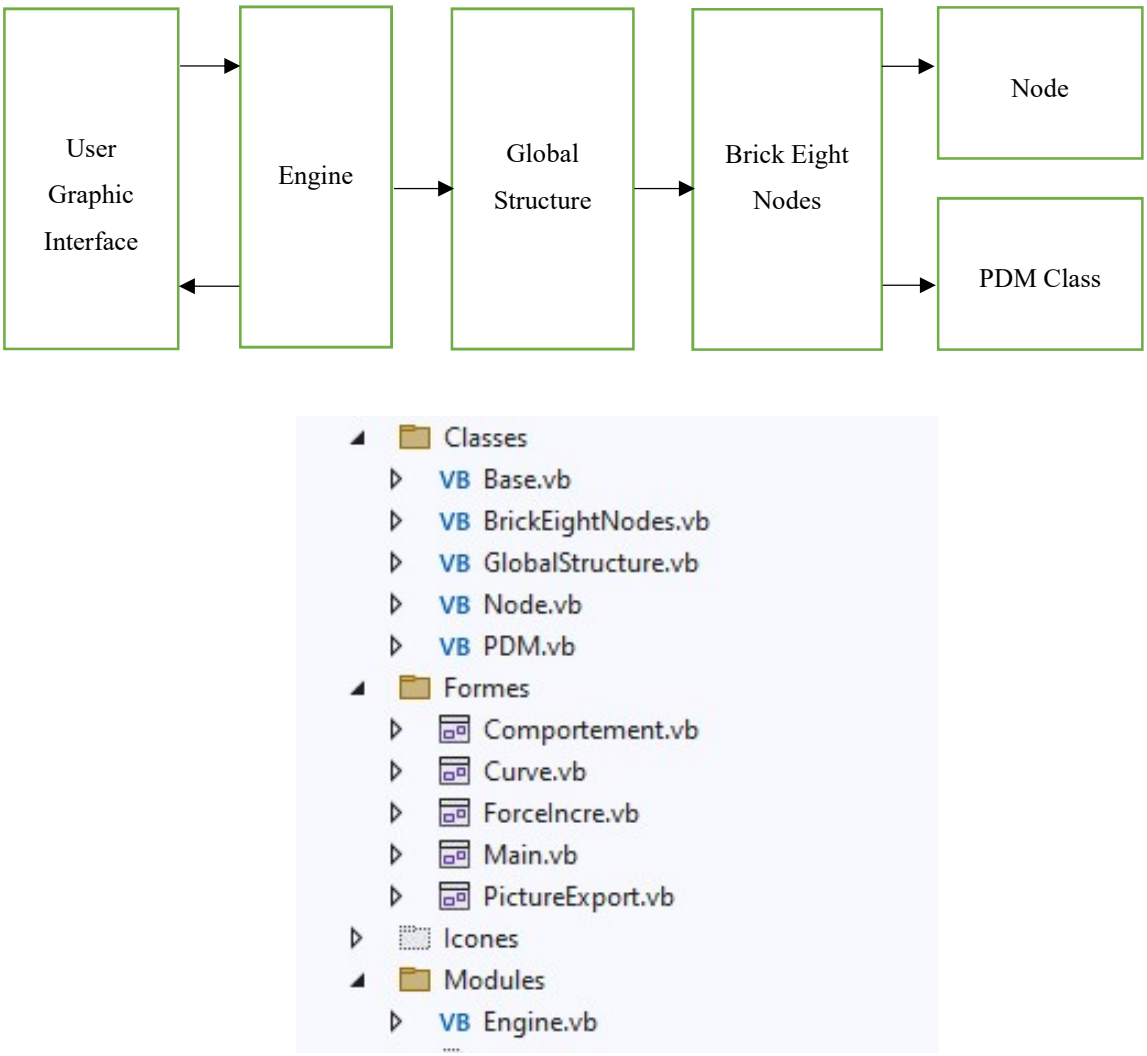


**Figure 3.2:** Concrete v2.0.0 architecture

## III.3 Eight nodes brick element

In order to discretize cubical and/or cylindrical concrete samples, a 3D element must be selected from various choices namely: the eight-node brick element (linear), the twenty-node brick element (parabolic), the six-node tetrahedron element (linear), and the ten-node tetrahedron element (parabolic). In the present work, the eight-node brick element (C3D8) was chosen for various logical reasons, which are: the displacement inside the element can be found through linear interpolation, and the low calculation cost of the C3D8 compared to the twenty-node brick element

In our code, the C3D8 element is fully integrated using 2x2x2 Gauss integration points. The node and the integration points are numbered following the convention of figure 3.3. The shape functions matrix of this element takes the following form:

$$[N]=\begin{bmatrix} N_v & 0 & 0 \\ 0 & N_u & 0 \\ 0 & 0 & N_w \end{bmatrix} \tag{3.1}$$

Where $N_u = N_v = N_w = \begin{bmatrix} N_1 & N_2 & N_3 & N_4 & N_5 & N_6 & N_7 & N_8 \end{bmatrix}$

$$\begin{cases} N_1(x,y,z)=\frac{1}{8}(1-x)(1-y)(1-z) \\ N_2(x,y,z)=\frac{1}{8}(1-x)(1+y)(1-z) \\ N_3(x,y,z)=\frac{1}{8}(1+x)(1+y)(1-z) \\ N_4(x,y,z)=\frac{1}{8}(1+x)(1-y)(1-z) \\ N_5(x,y,z)=\frac{1}{8}(1-x)(1-y)(1+z) \\ N_6(x,y,z)=\frac{1}{8}(1-x)(1+y)(1+z) \\ N_7(x,y,z)=\frac{1}{8}(1+x)(1+y)(1+z) \\ N_8(x,y,z)=\frac{1}{8}(1+x)(1-y)(1+z) \end{cases} \tag{3.2}$$

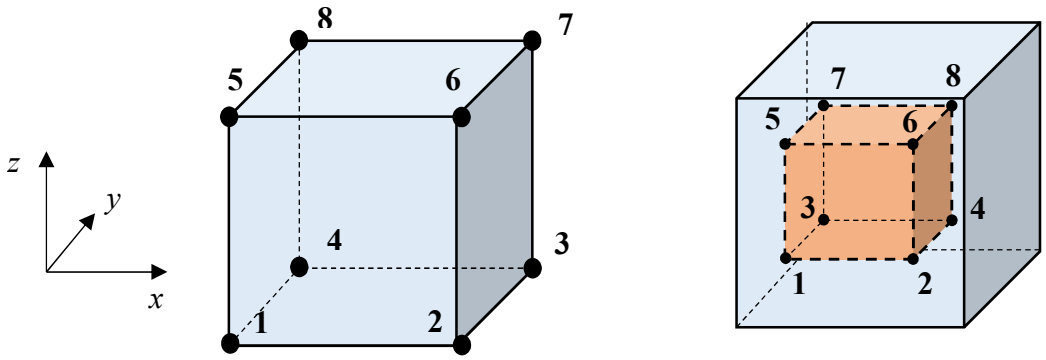


**Figure 3.3:** Eight nodes brick element (C3D8)

The stiffness matrix of this element is calculated according to:

$$[K]_e = \int_v [B]^T [D][B] dv \tag{3.3}$$

Where $[D]$ represents the stress-strain matrix, takes the following form:

$$[D] = \frac{E(1-\upsilon)}{(1+\upsilon)(1-2\upsilon)} \begin{bmatrix} 1 & \frac{\upsilon}{1-\upsilon} & \frac{\upsilon}{1-\upsilon} & 0 & 0 & 0 \\ \frac{\upsilon}{1-\upsilon} & 1 & \frac{\upsilon}{1-\upsilon} & 0 & 0 & 0 \\ \frac{\upsilon}{1-\upsilon} & \frac{\upsilon}{1-\upsilon} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\upsilon}{2(1-\upsilon)} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\upsilon}{2(1-\upsilon)} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\upsilon}{2(1-\upsilon)} \end{bmatrix} \tag{3.4}$$

And the matrix $[B]$ is calculated as follows;

$$[B]=[\partial][N] \text{ with } [\partial]=\begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{bmatrix} \tag{3.5}$$

## III.4 Mesh generation

As known, Mesh generation is the process of dividing the whole structure into a set of defined elements. Usually, for one-dimensional structures, each of bar or beam element with two or three nodes can be used to discretize the continuum. For 2D structures, both triangle and quadrilateral elements with three, four, six, and eight nodes can be used to generate the mesh. For our computer software, we need to discretize both cubical and cylindrical elements using the C3D8 element as mentioned previously.

In Concrete v2.0.0 , the meshing subroutines are affiliated to the Global structure class where two main subroutines can be found to ensure the meshing of both supported shapes namely the cylindrical shape and the cubical shape. The first subroutine called “GenerateMeshCylindre” ensures the meshing of the cylindrical shape. This subroutine goes through the 2D discretization of the cylinder section (circle) provided by the Triangle.Net library as illustrated in figure 3.4 to generate the 3D mesh. This subroutine executes the following steps:

- Calculate the cylinder section and generate the polygon that represents the section (circle)
- Generate 2D mesh of the section
- Delete the double quadrilaterals elements
- Build the levels for a given distance
- Generate the 3D mesh

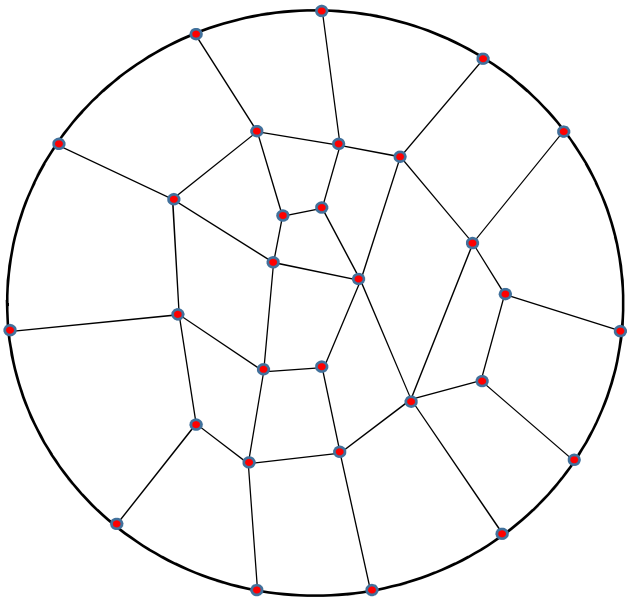

**Figure 3.4:** 2D mesh of circle shape

The second subroutine called “GenerateMeshCube” ensures the meshing of cubical shape. In the same way of GenerateMeshCylindre subroutine, this subroutine goes through the 2D discretization of the cubical section (rectangle) as illustrated in figure 3.5 to generate the 3D mesh. This subroutine follows the next instructions:

- Calculate the cubic section and generate the rectangle
- Using CalculatePointsCube subroutine, calculate the nodes list that is used to generate the 2D mesh of the rectangle
- Generate the 2D mesh
- Delete the double quadrilaterals elements
- Build the levels for a given distance
- Generate the 3D mesh

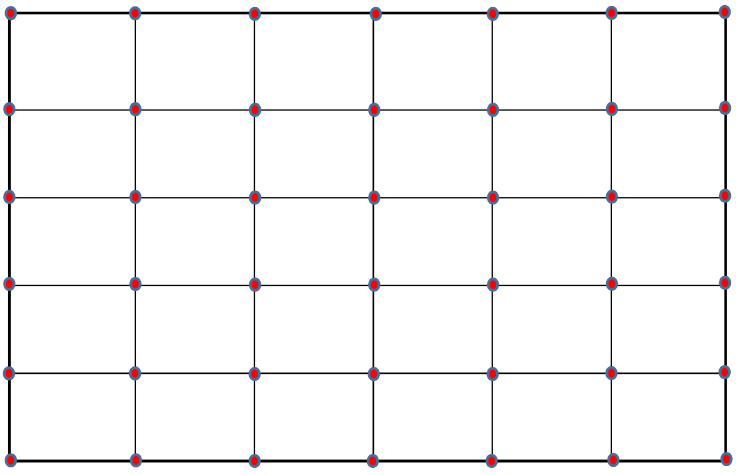

**Figure 3.5:** 2D mesh of rectangle shape

Both subroutines GenerateMeshCylindre and GenerateMeshCube are presented in the appendix.

## III.5 OpenTK library

The Open Toolkit is a set of fast, portable, low-level C# bindings for OpenGL, OpenGL ES, OpenAL, and OpenCL. It runs on all major platforms and powers hundreds of Apps, games, and scientific research programs. Practically, the OpenTk can be used on the multiple technologies such as WPF, WinForms, Xamarin, Avalonia, WinUI, and UWP.

Since OpenGL is a graphics API and not a platform of its own, it requires a language to operate in and the language of choice is C++. Therefore a decent knowledge of the C++ programming language is required to master OpenGL which is unfortunately unenviable in the .net Framework. Luckily, OpenTK gives us the possibility to operate OpenGL inside the .net Framework environment, which offers us the ability to draw the elements with a high level of speed and quality. The main subroutines that are used to draw 2D and 3D elements are delivered in the appendix

## III.6 PDM Class description

The PDM Class was developed as part of our academic finite element code “Concrete v2.0.0” in order to model damaged concrete structures. The design of this class is mainly based on several fields, functions; and subroutines that are required to implement the PDM. These fields are mainly used to evaluate the value of the yield function "F" and the value of the potential function "G". Therefore, each parameter used in Eqs (1.28), (1.22), (1.26), (1.27), and (1.30) has a correspondence filed in the PDM class (Table 3.1).

Also, it is essential to evaluate each of; the yield function, the derivatives of the yield function, the potential function, and the derivative of the potential function in order to identify the material behavior and the plastic strain (F<0 signified that the current integration point is elastic and F>0 signified that the plastic yielding is indicated and $\Delta\sigma$ is incorrect). Hence, the PDM Class includes the functions and the subroutines summarized in Table 3.2 (The code source of the PDM class is presented in the Appendix).

**Table 3.1:** Fields used in the PDM Class

| Field | Description |
|---|---|
| E | The initial undamaged stiffness E0. |
| v | Poisson Ratio $\upsilon$ |
| fb0_fc0 | The ratio of biaxial and uniaxial compressive yield strengths $f_{b0}/f_{c0}$ |
| Kc | The ratio of second stress invariants on tensile and compressive meridians $K_c$ |
| Excent | The flow potential eccentricity $\varepsilon$ |

| | |
|---|---|
| SigmaT0 | The uniaxial tensile stress at failure $\sigma_{t0}$. |
| PsiAngle | The dilation angle $\psi$ |
| fck | The compressive strength of concrete $f_{ck}$ . |
| ac | Dimensionless coefficients $a_c$, $a_t$, $b_c$, and $b_t$. |
| at | |
| bc | |
| bt | |
| SigmaC | Compressive stress value $\sigma_c$ for a given compressive inelastic strain |
| SigmaT | Tensile stress value $\sigma_t$ for a given tensile inelastic strain |
| dc | Compressive damage parameter value $d_c$ for a given compressive inelastic strain |
| dt | Tensile damage parameter value $d_t$ for a given tensile inelastic strain |
| d | Damage parameter value d |

**Table 3.2:** Functions and Subroutines used in the PDM Class

| Function / Subroutines Name | Description |
|---|---|
| YieldFunctionEstimation | This function returns with the value of the yield function for a given stress tensor. |
| DerivativeYieldFunctionStress | This function returns with the value of the derivative of the yield function with respect to the stress tensor for a given stress tensor. |
| DerivativeYieldFunctionStrain | This function returns with the value of the derivative of the yield function with respect to the compressive inelastic strain for a given stress tensor. |
| DerivativePotentialFunction | This function returns with the value of the derivative of the potential function for a given stress tensor |
| DamageParametres | This subroutine calculates according to Alfarah approach, Bakhti approach, and user data approach the following parameters:<br>- The tensile stress for a given inelastic strain;<br>- The compressive stress for a given inelastic strain;<br>- The tensile damage parameter for a given inelastic strain;<br>- The compressive damage parameter for a given inelastic strain.<br>- The damage parameter |
| DLambda | This function returns with the value of the plastic multiplier. |
| ParamatersCDP | This function is inherited from Base class and returns with the values of:<br>- Alfa parameter<br>- Beta Parameter<br>- Gamma parameter |
| FindInelasticStrain | This function returns with the inelastic strain value. |
| PlasticStressImplicit | This function returns with the plastic strain tensor. |

## III.7 Concrete V2.0.0 description

Concrete v2.0.0 is a Windows App developed under visual studio using vb.Net coding language. The OOP technology was used in order to improve the design quality of our computer code. The first version of this code was developed only to model cylindrical and cubical elements considering the concrete as elastic liner material. Otherwise, considering the concrete material as linear elastic

material provides inaccurate results and shows a major weakness regarding the degradation process of concrete. To overcome these issues, the second version was developed to simulate the real behavior of concrete and improve the predicting of the concrete degradation in both cases tension and compression. In the user interface graphic, two main tabs are available, the first one is the Geometry tab, where the following tasks are available:

- List of buttons number “1“is used to choose the shape of the concrete sample. Two shapes are available, the cubical one and the cylindrical one (Illustrated in figure 3.6)
- List of buttons number “2” allows the user to input the shape dimensions. For the cubical case, the user can input the length; the width, and the height of the sample. For the cylindrical shape, each of; the height and the diameter must be entered. In addition, the drawing factor must be inputted. (Illustrated in figure 3.6)
- List of buttons number “3” provides the ability to reinforce the concrete by adding layers of composite materials to the concrete sample. This feature will be available in the third version of Concrete software. (Illustrated in figure 3.6)
- Button “4” is used to generate the mesh (Figure 3.7)
- Button “5” allows the user to broaden an existing mesh. This button cannot be used before generating the mesh. (Figure 3.7)
- Button “6” allows the user to squeeze an existing mesh. This button cannot be used before generating the mesh. (Figure 3.7)
- List of buttons number “7” allows the user to select the drawing mode namely the 3D model or the 2D model (Figure 3.8)

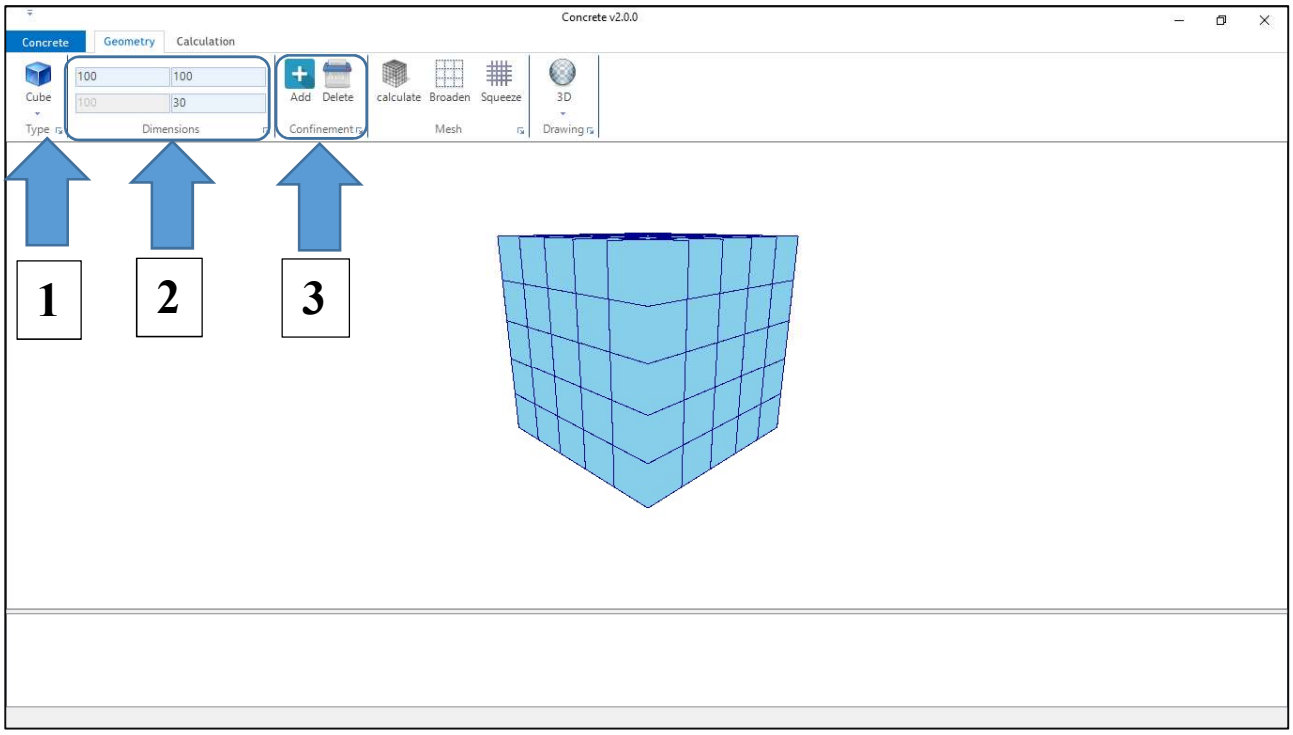


**Figure 3.6:** Concrete v2.0.0. Screenshot N01

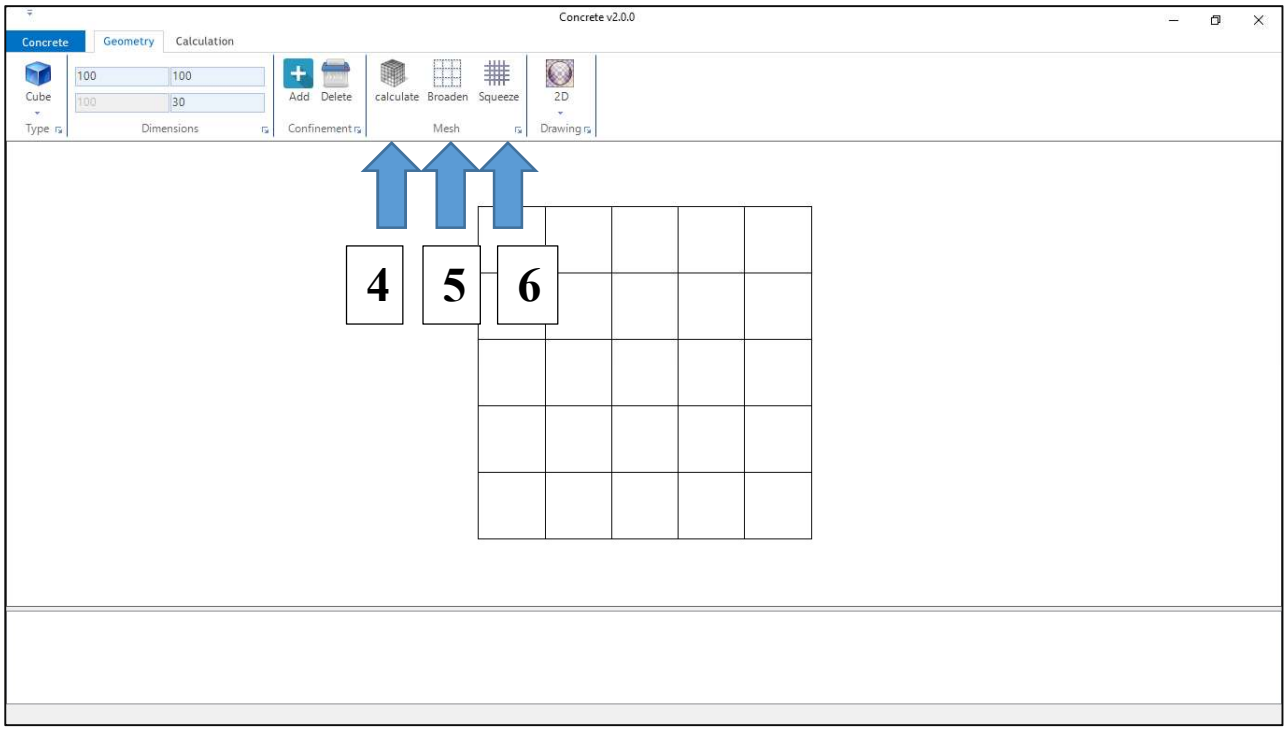


**Figure 3.7:** Concrete v2.0.0. Screenshot N02

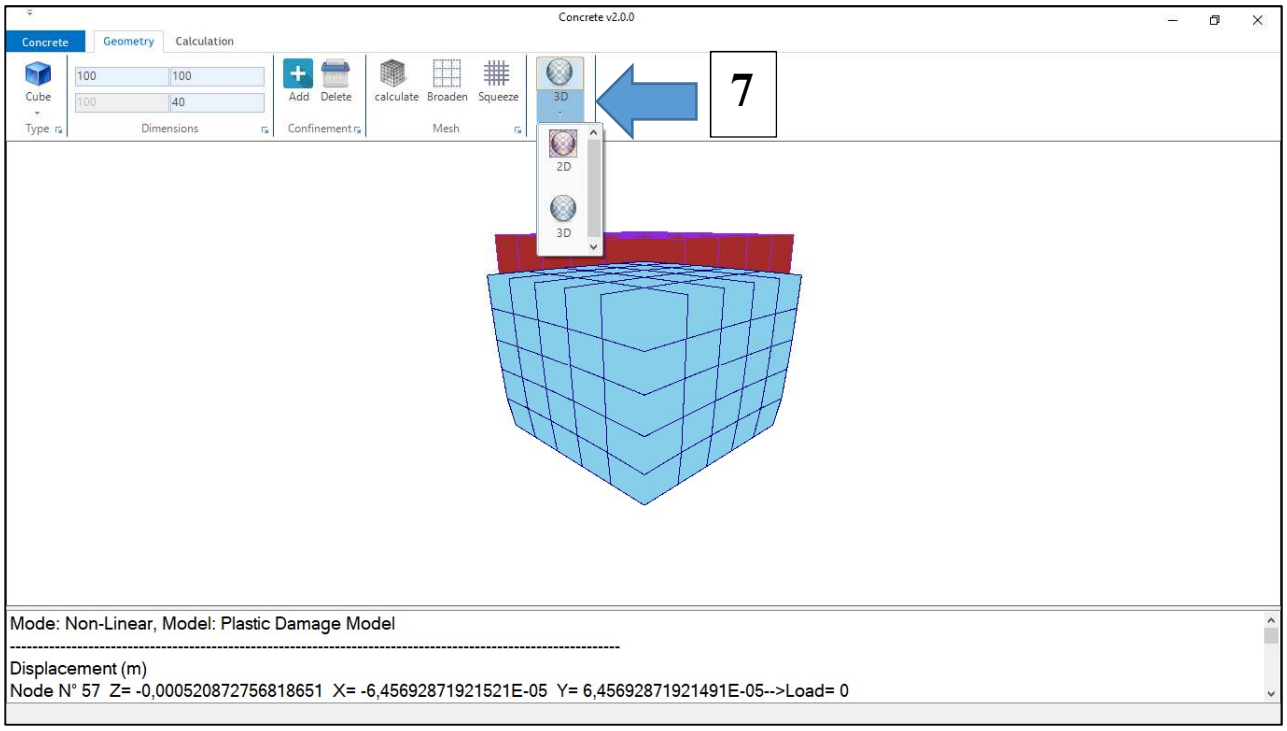


**Figure 3.8:** Concrete v2.0.0. Screenshot N03

The second tab available in our computer code is the Calculation tab, where the linear and the nonlinear calculations (using PDM) are available. In addition to the output button, the following tasks are available in the calculation tab:

- List of buttons number “8” is used to model concrete samples for linear analysis. The button calculation is used to start the calculation process (Figure 3.9). The following parameters are required to perform the calculation:
    - The Young modulus
    - Poisson’s ratio.
- List of buttons number “9” is used to model concrete samples for nonlinear analysis using the plastic damage model. The user can choose the compressive case or the tensile case. Also, the user can input the compressive stress strength via the textbox $f_{cm}$ (Figure 3.10). Default values are suggested for the following parameters:
    - $f_{b0} / f_{c0}$ The ratio of biaxial and uniaxial compressive yield strengths (default value equal to 1.16)

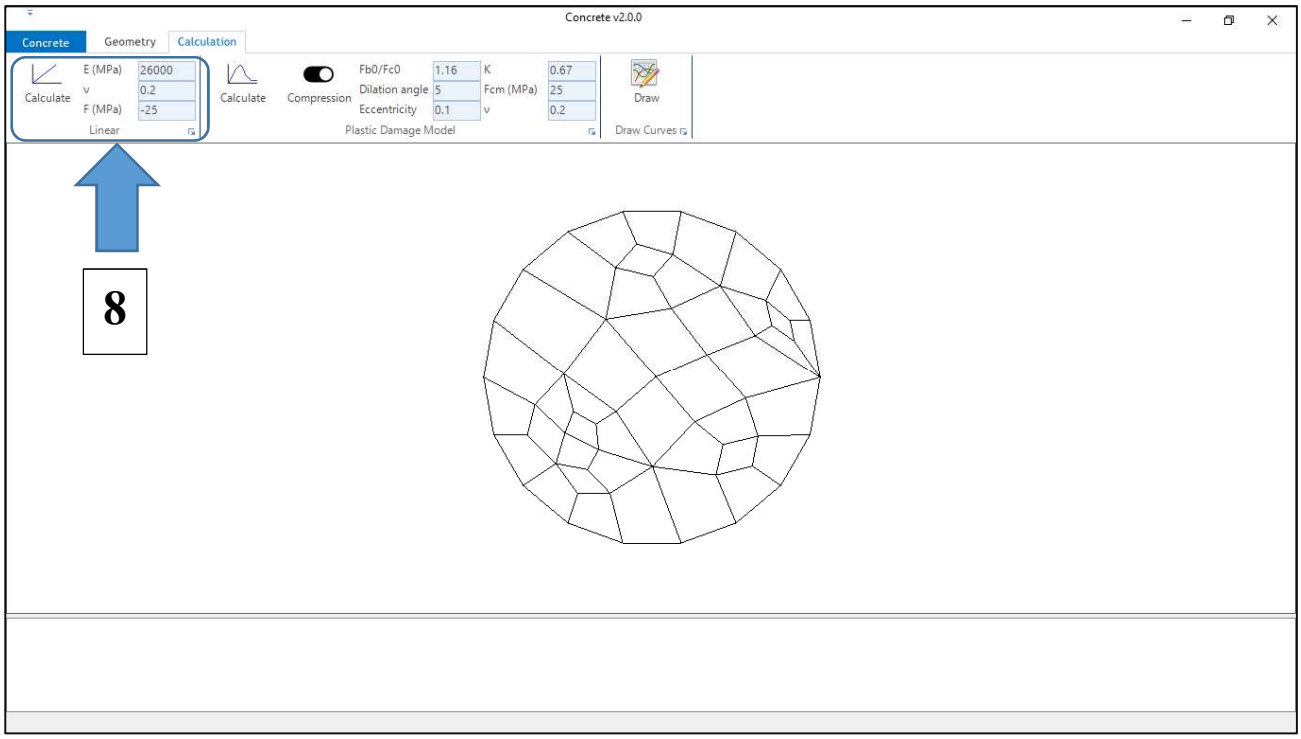


**Figure 3.9:** Concrete v2.0.0. Screenshot N04

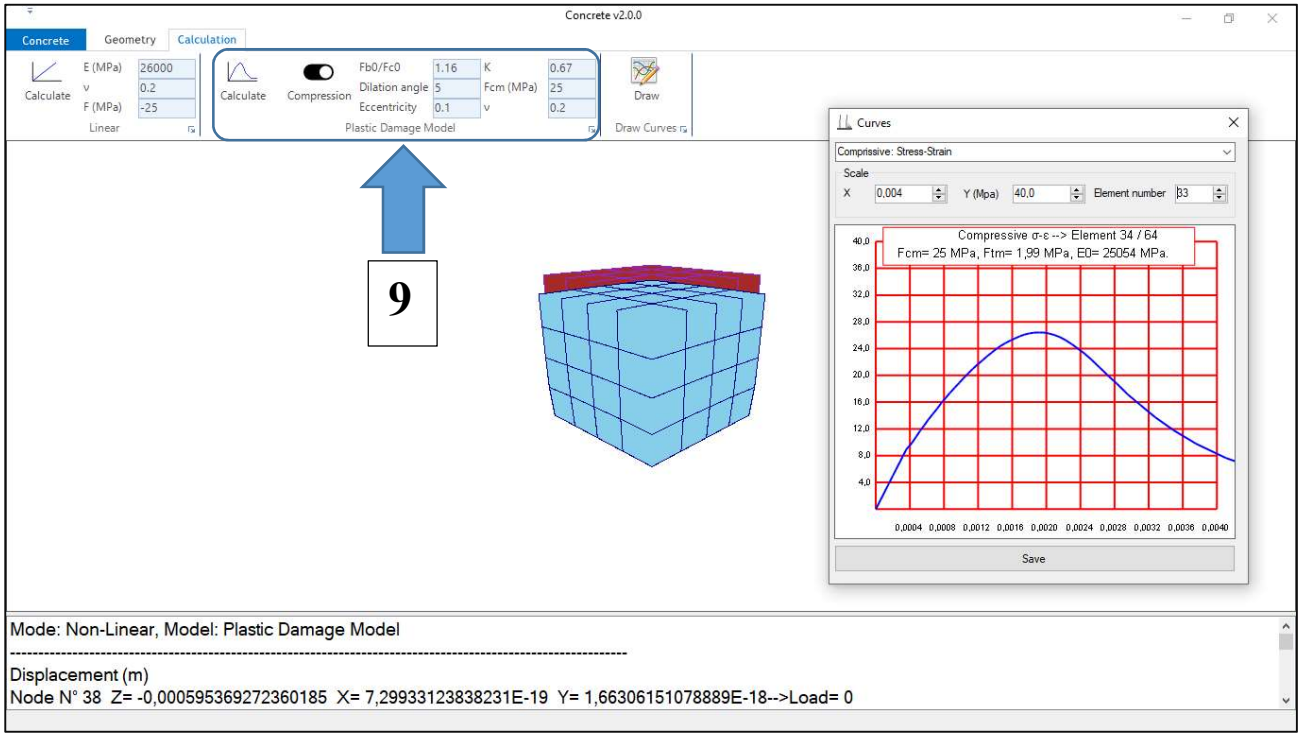


**Figure 3.10:** Concrete v2.0.0. Screenshot N05

- $K_c$ The ratio of second stress invariants on tensile and compressive meridians (default value equal to 0.67)

- $\varepsilon$ The flow potential eccentricity (default value equal to 0.1)
- $\psi$ The dilation angle (default value equal to 5 degrees)
- Poisson’s ratio (default value equal to 0.2)

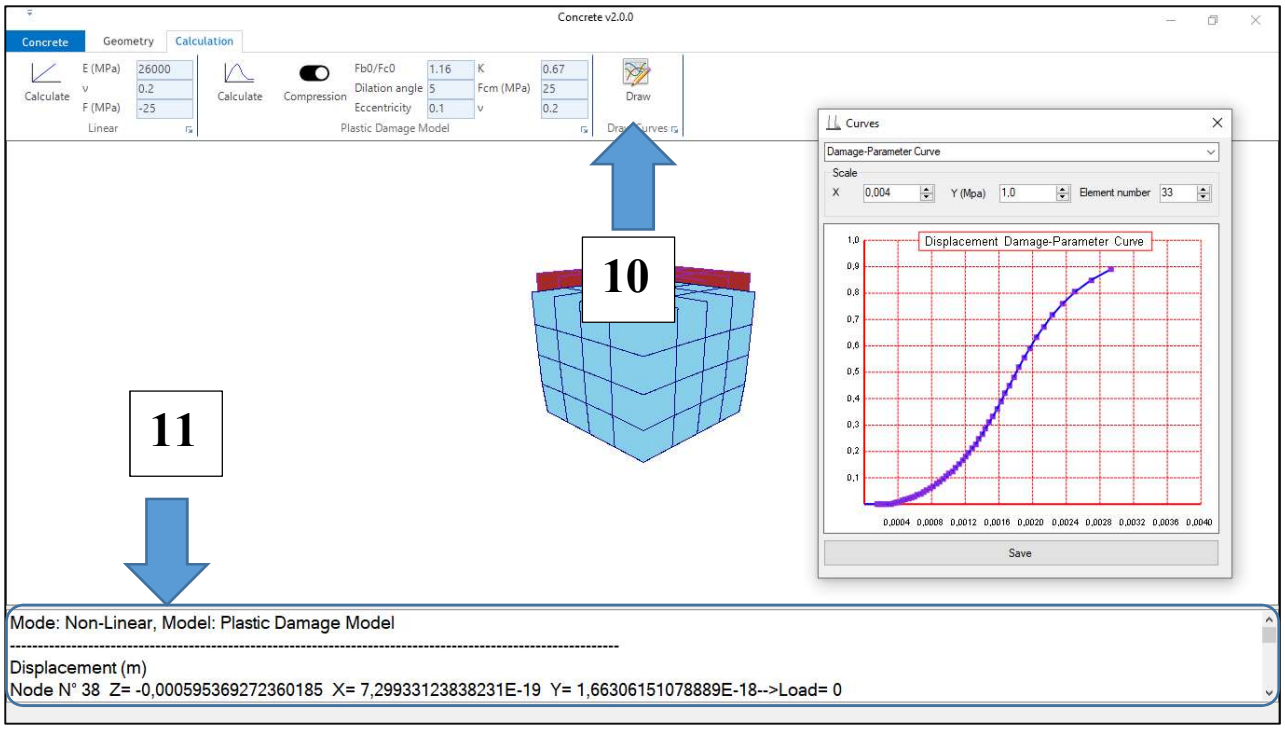


**Figure 3.11:** Concrete v2.0.0. Screenshot N06

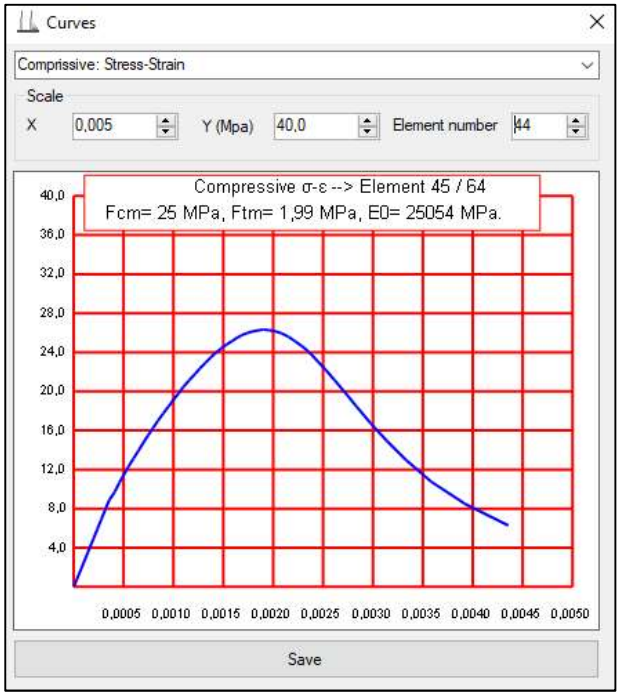


**Figure 3.12.a:** Concrete V2.0.0 outputs - Compressive stress-strain curve

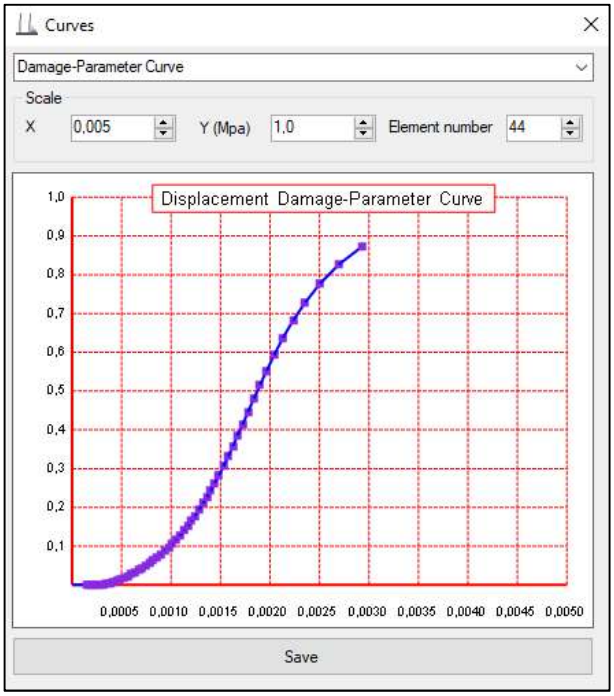


**Figure 3.12.b:** Concrete V2.0.0 outputs - Damage parameter curve

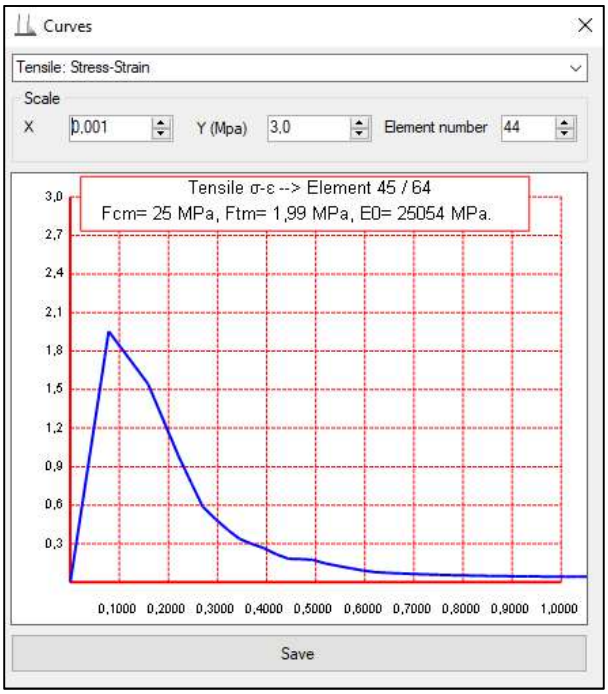


**Figure 3.12.c:** Concrete V2.0.0 outputs - Tensile stress-strain curve

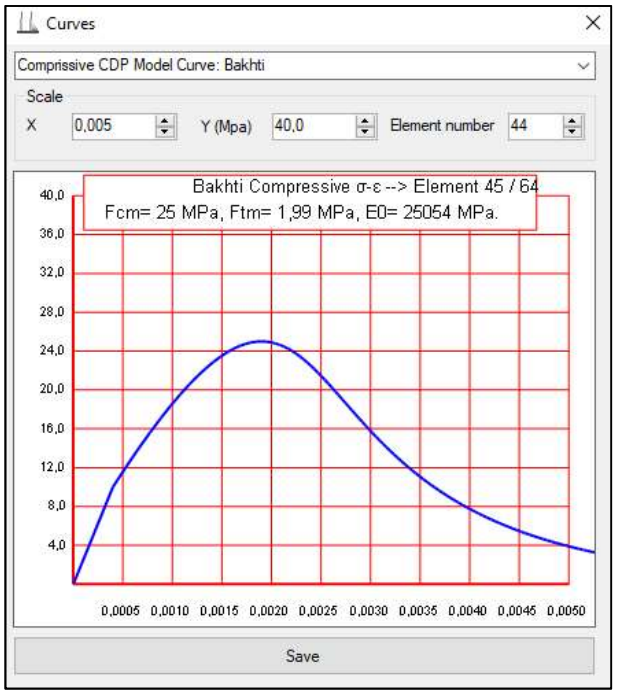


**Figure 3.12.d:** Concrete V2.0.0 outputs - Compressive stress-strain (BAKHTI)

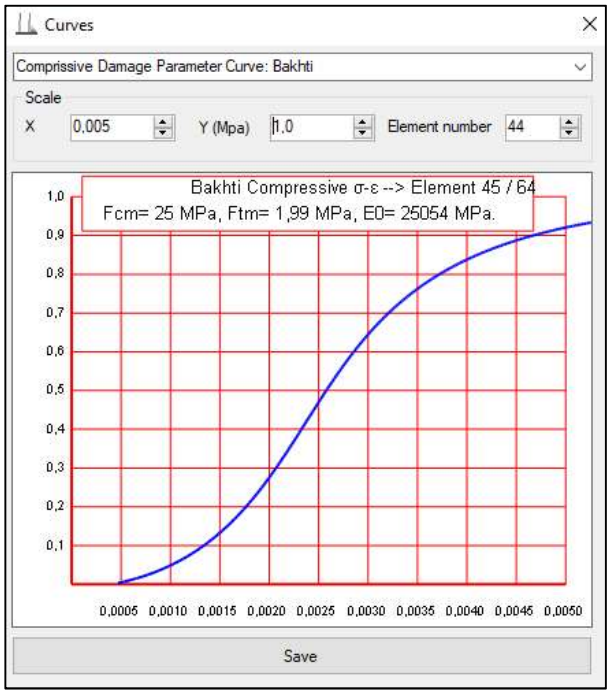


**Figure 3.12.e:** Concrete V2.0.0 outputs - Compressive damage parameter curve (BAKHTI)

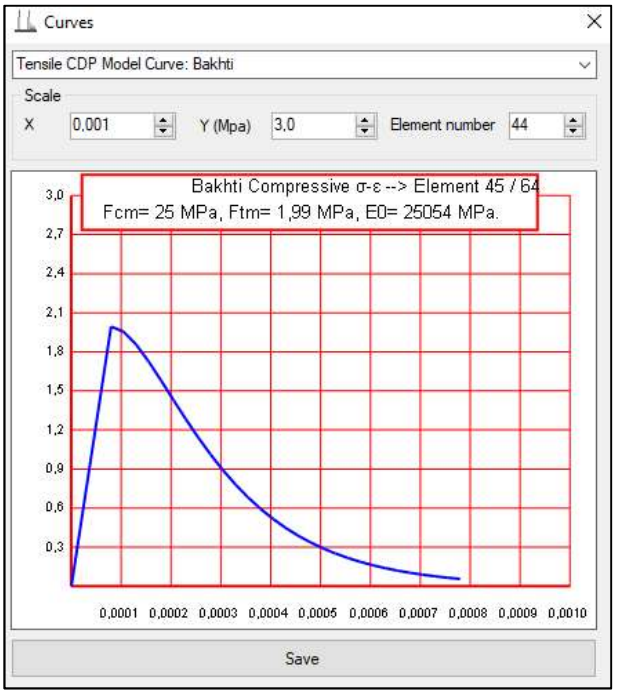


**Figure 3.12.f:** Concrete V2.0.0 outputs - Tensile stress-strain curve (BAKHTI)

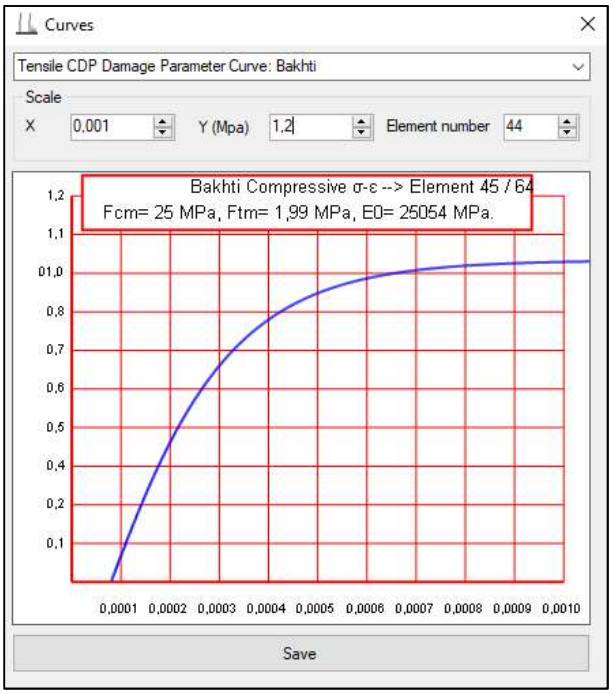


**Figure 3.12.g:** Concrete V2.0.0 outputs - Tensile damage parameter curve (BAKHTI)

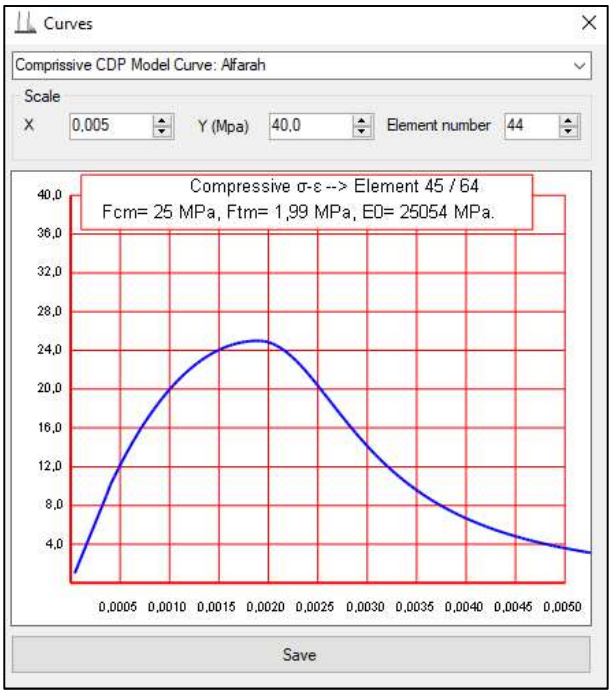


**Figure 3.12.h:** Concrete V2.0.0 outputs - Compressive stress-strain (Alfarah)

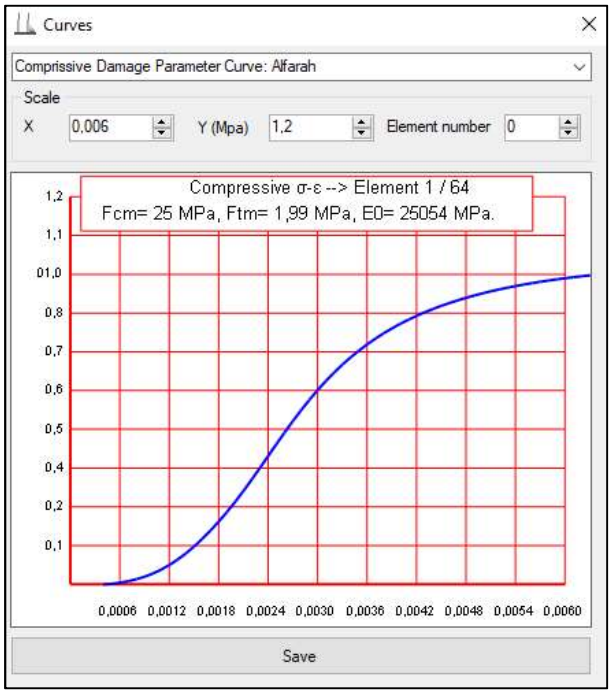


**Figure 3.12.i:** Concrete V2.0.0 outputs - Compressive damage parameter curve (Alfarah)

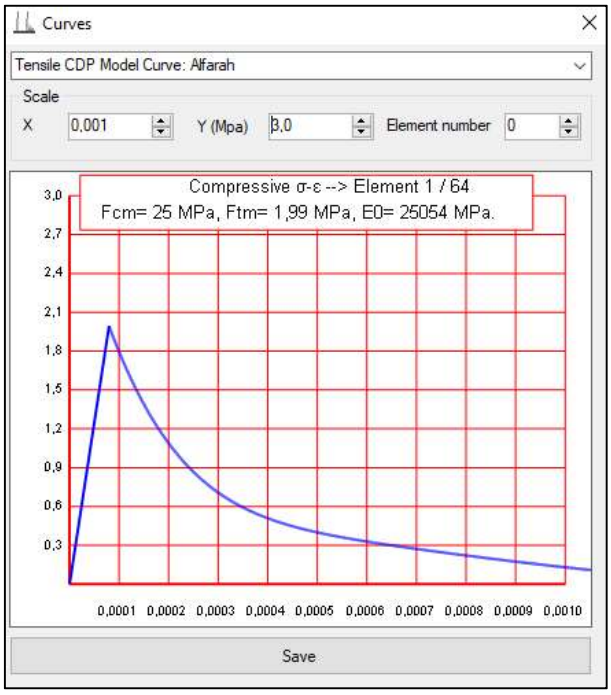


**Figure 3.12.j:** Concrete V2.0.0 outputs - Tensile stress-strain curve (Alfarah)

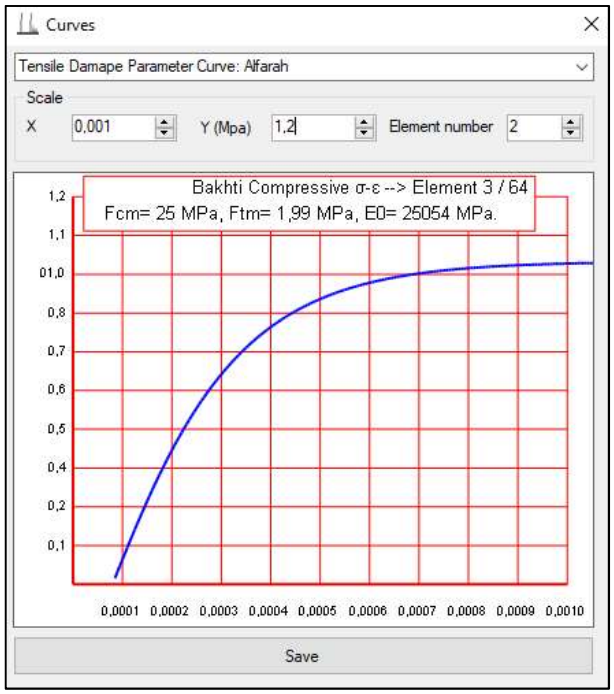


**Figure 3.12.k:** Concrete V2.0.0 outputs - Tensile damage parameter curve (Alfarah)

- Button “10” allows the user to display the following curves (Figure 3.11):
    - Compressive stress-strain (Figure 3.12.a)
    - Damage-parameter curve (Figure 3.12.b)
    - Tensile: stress-strain (Figure 3.12.c)
    - Compressive DPM Model Curve according to the author’s approach (Figure 3.12.d)
    - Compressive Damage Parameter Curve: Bakhti (Figure 3.12.e)
    - Tensile CDP Model Curve: Bakhti (Figure 3.12.f)
    - Tensile CDP Damage Parameter Curve: Bakhti (Figure 3.12.g)
    - Compressive CDP Model Curve: Alfarah (Figure 3.12.h)
    - Compressive Damage Parameter Curve: Alfarah (Figure 3.12.j)
    - Tensile CDP Model Curve: Alfarah (Figure 3.12.j)
    - Tensile Damage Parameter Curve: Alfarah (Figure 3.12.k)
- List box “11” displays the displacements of each node (Figure 3.11).

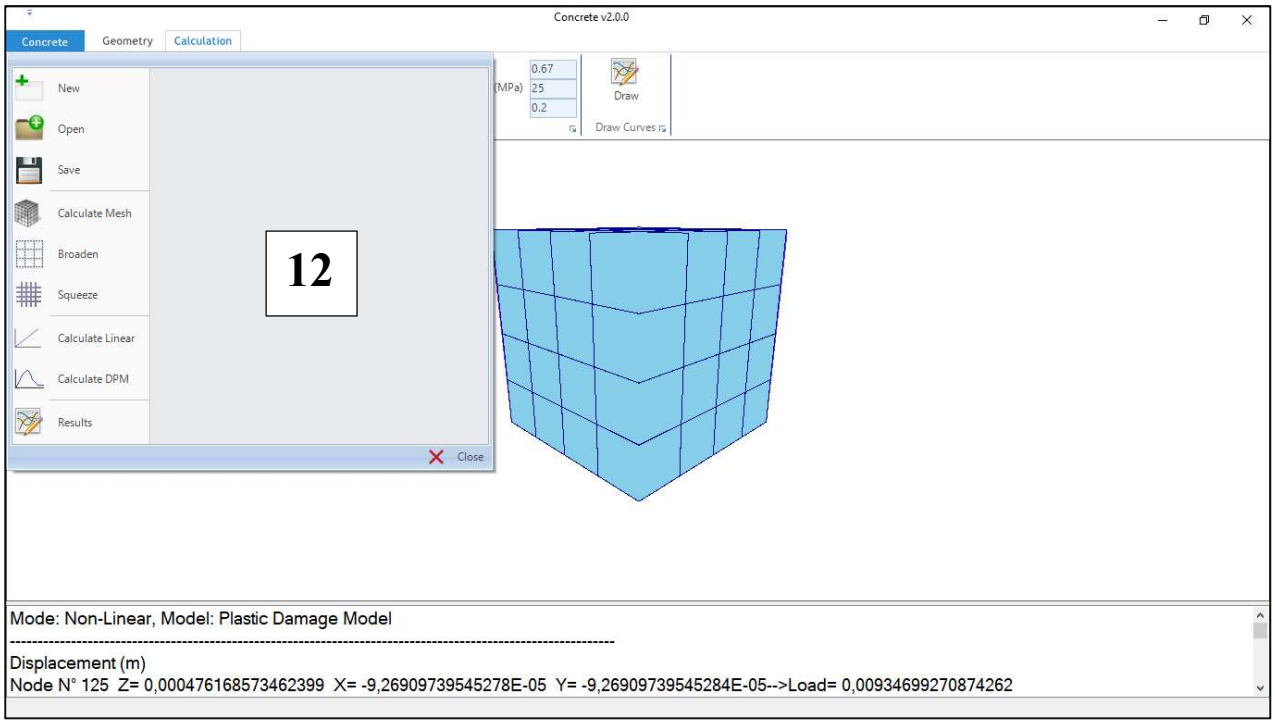


**Figure 3.13:** Concrete v2.0.0. Screenshot N07

- Menu “12” (Figure 3.13) allows the user to create a new project, save and open an existing project and a quick access for:
    - Generate mesh
    - Broaden an existing mesh
    - Squeeze an existing mesh
    - Model concrete samples for linear analysis
    - Model concrete samples for non-linear analysis using PDM
    - Display results

## III.8 Conclusion

In this chapter, a full description of the new computer code “Concrete v2.0.0” was provided. The used coding paradigm was described in this chapter where the main advantage is improving the design quality of our computer code. In addition, the architecture of Concrete v2.0.0 was provided where the developed code is based on four classes which are the “Global Structure” class, “Brick Eight Nodes” class, “Node” class, and the “PDM” Class. Also, the used element was delivered where the closed forms of each; the shape functions, the stress-strain matrix, and the stiffness matrix were provided. The mesh generation process of the cylindrical shapes and the cubical shapes was described in this

chapter, in addition to several subroutines and functions that are employed in our software, each of; the manual calculations of the 2D meshing and the Triangle.Net calculation were also provided. In the same manner, the drawing process through the OpenTK library was presented in this chapter, where multiple subroutines are presented.

A full description of the PDM Class was delivered as part of our academic finite element code “Concrete v2.0.0”. The design of this class is mainly based on multiple required fields, functions; and subroutines to implement the plastic damage model. These fields, functions, and subroutines are delivered in tables 3.1 -3.2. In the same manner, the user interface graphic of “Concrete v2.0.0” was described in this chapter where all the required tools/buttons in our computer code were demonstrated, the reader can easily run the software simply by following the delivered description of the software.

# Chapter IV : Investigation of the inputs and the outputs of “Concrete v2.0.0”

## IV.1 Introduction

In order to validate our computer software, a comparative study between the developed approach (proposed approach for computing the stress-strain diagrams and the damage parameters evolution) and stress-strain curves from the literature is provided in this chapter in addition to comparing the outcomes of Concrete v2.0.0 with experimental evidence and analytical approaches. Also, this chapter provides numercical study to examine the mesh sensitivity. For the auto estimation of the stress-strain curves, the outcomes of our approach were compared with the following solutions:

- The experimental stress-strain curves of Mohamad Ali et al [50],
- The stress-strain curves generated according to Alfarah correlations [23],
- The stress-strain curves generated according to Thorenfeldt correlations [102].

Furthermore, a comparison was preseted between the compressive and tensile stress-strain curves generated by “Concrete v2.0.0”with multiple compressive and tensile stress-strain correlations from the literature. For the compressive case, five values of the concrete compressive strength were selected (20, 25, 30, 35, and 40 MPa), for comparing with the stress-strain correlations of :

- Lubliner [26],
- Desayi and Krishan [47],
- Kratzig and Polling [29],

For the tensile case, same values of compressive strength were used for comparing with the outcomes of Lubliner correlations [26] and the outcomes of Thorenfeldt correlations [102]. The mesh sensitivity was examined through three cases of the mesh densities which are; 27 elements, 64 elements, and 125 elements (for a cubic sample with the following dimensions: 250 mm long, 250 mm wide, and 250 mm high).

For the compressive case, the outcomes of Concrete v2.0.0 were compared with the experimental data provided by Watanabe et al [103] and with the closed-form solution suggested by Kratzig and Polling [29]. For the tensile case, the outcomes of Concrete v2.0.0 were compared with experimental data provided by B. Ahmed et al [32] and with the outcomes of the Thorenfeldt approach [102].

## IV.2 Validation of the proposed approach for computing the stress-strain diagrams and the damage parameters evolutions

In order to validate the proposed approach for computing the stress-strain curves and the damage parameters evolutions, the stress-strain curves generated by “Concrete v2.0.0” according to the

present approach were compared with experimental results for the compression case and with closed-form solutions for the tension case, the results are illustrated in figures 4.1, 4.1, 4.3, and 4.4.

Figure 4.1 shows the compressive stress-strain curves of experimental tests provided by Mohamad Ali et al [50] and the outcomes of the proposed approach for different compressive strengths $f_{ck}$ (8.7, 17.3, 19.7 and 24 MPa). From this figure, it is observed that for every sample, both curves are very close, which proves the efficiency of the presented approach.

Figures 4.2, 4.3 and 4.4 show the tension stress-strain curves generated by "Concrete v2.0.0" (according to the proposed algorithm) and the outcomes of the analytical solution of Alfarah [23] and the analytical solution of Thorenfeldt [102]. The tension stress-strain curves generated according to Alfarah formulas were calculated for a cubic sample with the following dimensions: 100 mm long, 100 mm wide and 100 mm high.

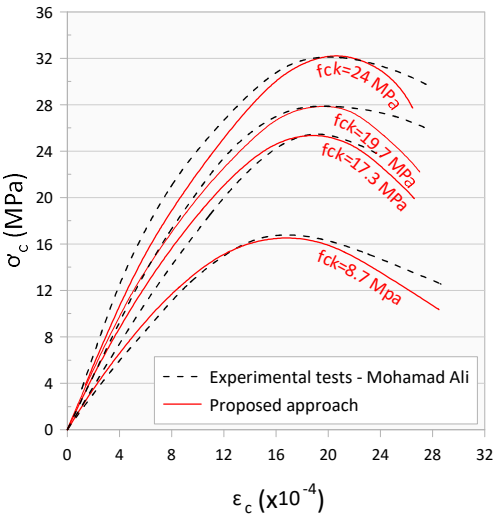


**Figure 4.1:** Validation of the auto-estimation of the compressive stress-strain curves

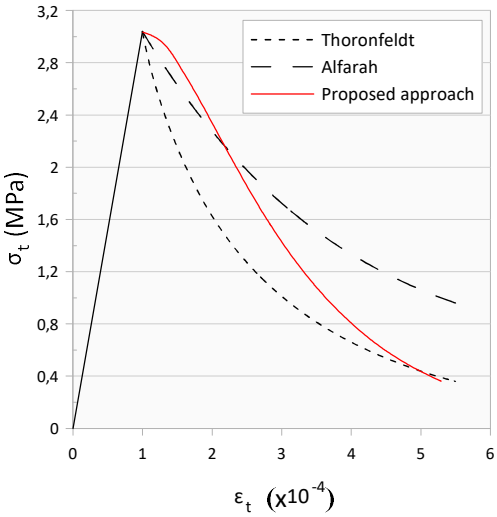


**Figure 4.2:** Validation of the auto-estimation of the tensile stress-strain curve, $f_{ck}$=32 MPa

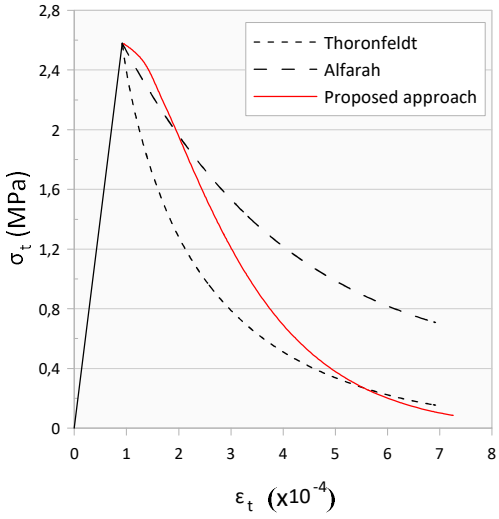


**Figure 4.3:** Validation of the auto-estimation of the tensile stress-strain curve, $f_{ck}$=25 MPa

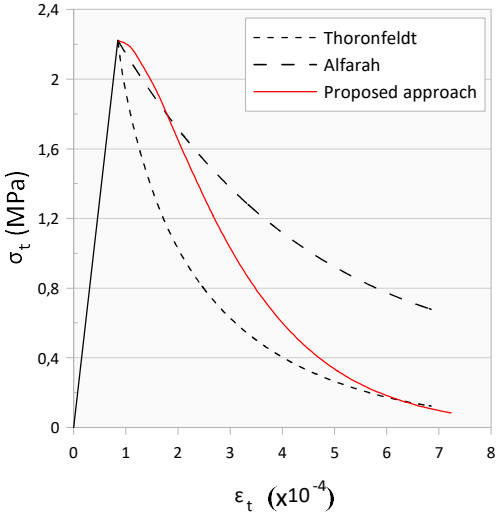


**Figure 4.4:** Validation of the auto-estimation of the tensile stress-strain curve, $f_{ck}$=20 MPa

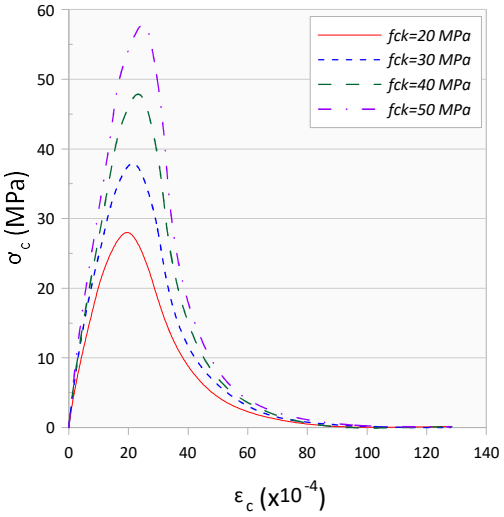


**Figure 4.5:** Compressive Stress-strain curves for different compressive strength

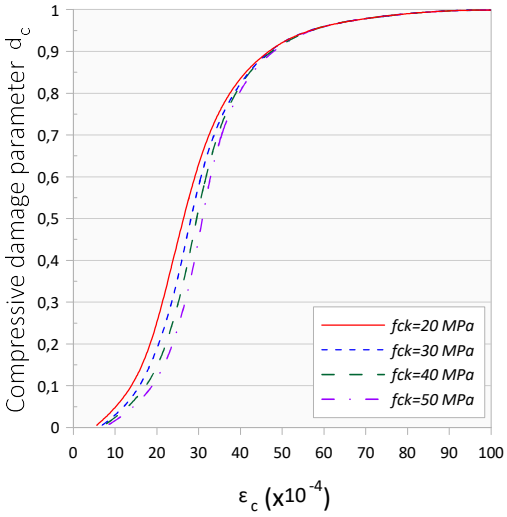


**Figure 4.6:** Compressive damage parameter evolution

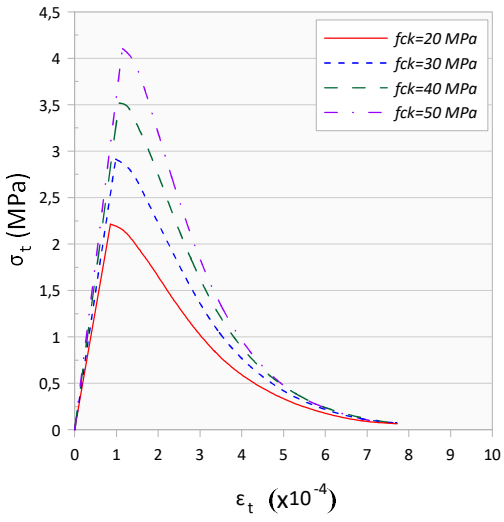


**Figure 4.7:** Tensile Stress-strain curves for different compressive strength

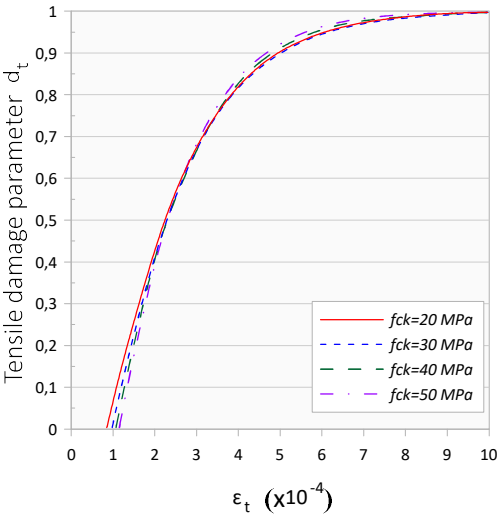


**Figure 4.8:** Tensile damage parameter evolution

The tension stress-strain curves generated according to the approach of Thorenfeldt are linear up to the uniaxial tensile strength, then determined by:

$$\sigma_t = f_{tm}\left(\frac{\varepsilon_{tm}}{\varepsilon_t}\right)^{\left(0.7+1000\varepsilon_t\right)} \tag{4.1}$$

Figures 4.2, 4.3, and 4.4 show that the proposed approach gives similar results to Alfarah approach at the beginning, then, they start to move gradually away until they reach the Thorenfeldt outcomes at the end of the curve. Figures 4.5, 4.7 illustrate the compressive and the tensile stress-strain diagrams generated according to the proposed approach.

Figures 4.6, 4.8 illustrate the compressive and the tensile damage parameters evolution generated according to the present approach. The curves were generated for four concrete strengths $f_{ck}$ (20, 30, 40, and 50 MPa).

Figure 4.5 illustrates compressive strength effect on the compressive stress-strain curves. The maximum values increased with the augmentation of the compressive strength, while the effect of compressive strength on the compressive damage parameter evolution is negligible for compressive strain $\varepsilon_C$ higher than 60 x$10^{-4}$.

From Figure 4.7, it is observed that the curves can be divided into three parts. The curves are quite close to each other in the first part which represents the linear behavior of concrete, until they reach the peak points, where the maximum differences values are obtained. Then the curves begin to get closer to each other in the second part, until they assemble in the third part. Similarly, the tensile curves presented in Figure 4.8 consist of three parts. In the first part, it is observed that the curves begin converging and reach a point of intersection. In the second part, it is noticed that the curves begin to converge and reach the maximum values, then started to get closer to each other until reaching the third part where the curves assemble. Moreover, it is observed that the concrete strength influence on the tensile damage parameter evolution is very limited.

The stress-strain diagrams and the damage parameters evolution generated according to the developed approach are completely independent of the mesh size ( $L_{eq}$ ) value. This conclusion is based on the fact that the stresses and the damage parameters values are mainly correlated to the coefficients $a_c$, $a_t$, $b_c$, and $b_t$ according to Eq (1.42) and Eq (1.60), respectively. Furthermore, these coefficients are evaluated without the mesh size value (as demonstrated in figure 2.10).

## IV.3 Investigation of "Concrete v2.0.0" outcomes

This section aims to compare the compressive stress-strain and the tensile stress-strain curves generated by our computer code with multiple compressive and tensile stress-strain correlations from the literature. For the compressive case, five values of concrete strengths (20, 25, 30, 35, and 40 MPa) are selected to compare the outcomes of Concrete v2.0.0 with three stress-strain correlations for the compressive case namely:

- Desayi and Krishan [47] (Section I.5),
- Kratzig and Polling [29] (Section I.6.1.2.b),
- Lubliner et al [26] (Section I.6.1.2.b).

For the tension case, Lubliner et al [26] and Thorenfeldt et al [102]stress-strain correlations are carefully chosen to validate the outcomes of "Concrete v2.0." for the same values of concrete strengths. Young's modulus and the strain value at peak stress for each value of concrete strength are delivered in Table 4.2.

For Kratzig and Polling [29] correlation, all curves are calculated based on mesh size equal to 300 mm. The Model Code Recommendations [33] was used to calculate the following parameters:

- The initial tangent modulus of deformation of concrete $E_{ci} = 10000\, f_{cm}^{1/3}$ (4.2)
- The undamaged modulus of deformation $E_0 = E_{ci}\left(0.8 + 0.2\frac{f_{cm}}{88}\right)$ (4.3)
- The compressive stress that corresponds to zero crushing $f_{c0} = 0.4 f_{cm}$ (4.4)
- The value of the strain at the peak stress $\varepsilon_{cm} = 0.5\, f_{cm}^{0.31} \le 2.8\, x10^{-3}$ (4.5)
- The crushing/fracture energy (N/mm)

$$G_{ch} = \left(\frac{f_{cm}}{f_{tm}}\right)^2 G_F \text{ where } G_F = 0.073\, f_{cm}^{0.18} \quad (4.6)$$

Table 4.2 summarizes the values of the peak tensile stress and the strain at the peak tensile stress for each value of concrete strength that are required to evaluate the stress value according to Thorenfeldt correlation. For Lubliner correlations, we used the values of coefficients $a_c, a_t, b_c$ and $b_t$ that were calculated by Bakhti et al [104] and presented in Table 2.2 and Table 2.3. Also, according to this approach, the compressive and tensile stress-strain curves were divided into two parts. The first one represents the linear segment where the stress is evaluated according to Hooke's law. The second one represents the non-linear part where the stress is evaluated according to Lubliner's formulas.

Figures 4.9-4.13 present the compressive stress-strain curves generated by Concrete v2.0.0 together with the stress-strain curves generated according to the correlations of Lubliner, Desayi and Krishan, and Kratzig and Polling. From these figures, the following observations can be outlined:

- The curves generated by Concrete v2.0.0 are completely in harmony with the stress-strain curve generated according to Lubliner Formula, this observation can be justified by the fact that the used hardening function in the implementation of PDM is identical to Lubliner formulas.
- For the concrete strengths less than 25 MPa, The outcomes of “Concrete v2.0.0” are partially in harmony with the stress-strain curve generated according to Kratzig formula. For values more than 25MPa and by using the decomposition of Kratzig, we observed that the curves of Concrete v2.0.0 move away from Kratzig curve depending on the concrete strengths, especially in the third part.
- Using the decomposition of Kratzig, the outcomes of “Concrete v2.0.0” are partially in harmony with the stress-strain curve generated according to Desayi formula in the first and the second parts. For the third part, the difference between both curves is significant.

Figures 4.14-4.18 present the tensile stress-strain curves generated by “Concrete v2.0.0” together with the stress-strain curves generated according to the correlations of Lubliner and Thorenfeldt. From these figures, the following notes can be outlined:

- The outcomes of “Concrete v2.0.0” are in harmony with the stress-strain curve generated according to Lubliner formula which can be justified by the adopted hardening function which is the Lubliner formula.
- Using the decomposition of the tensile stress-strain curve illustrated in figure 2.8, the tensile stress-strain curves of “Concrete v2.0.0” are completely in harmony with the stress-strain curve generated according to Thorenfeldt formula in the first part. For the second part, Concrete v2.0.0 gives results partially similar to Thorenfeldt formula.

**Table 4.1:** input data of Desayi and Krishan curve

| $f_{cm}(MPa)$ | $\varepsilon_p(x10^{-3})$ | $E(MPa)$ |
|---|---|---|
| 20 | 1.8 | 22890 |
| 25 | 1.9 | 26130 |
| 30 | 2 | 29910 |
| 35 | 2.1 | 32890 |
| 40 | 2.2 | 35940 |

**Table 4.2:** input data of Thorenfeldt curve

| $f_{cm}(MPa)$ | $\varepsilon_{tm}(x10^{-4})$ | $f_{tm}(MPa)$ |
|---|---|---|
| 20 | 0.68 | 1.58 |
| 25 | 0.79 | 1.99 |
| 30 | 0.87 | 2.37 |
| 35 | 0.93 | 2.71 |
| 40 | 0.98 | 3.04 |

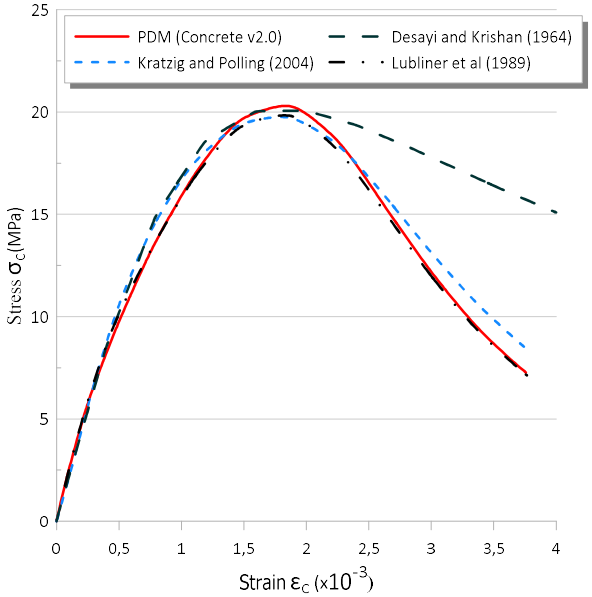


**Figure 4.9:** Compressive stress-strain curve for $f_{cm}$ = 20MPa

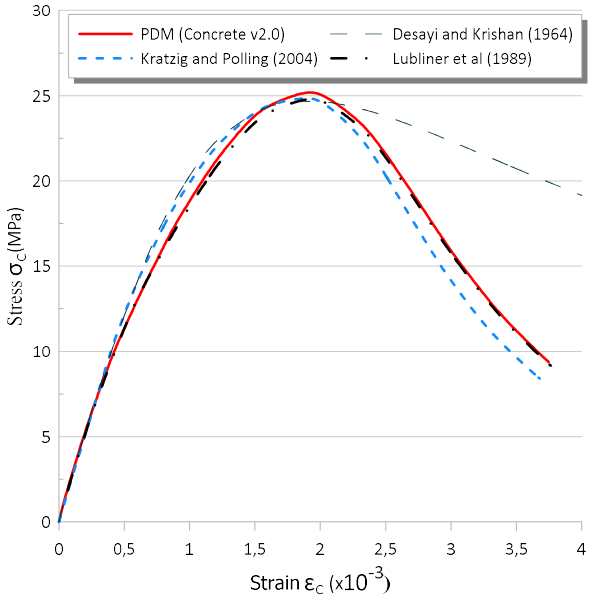


**Figure 4.10:** Compressive stress-strain curve for $f_{cm}$ = 25MPa

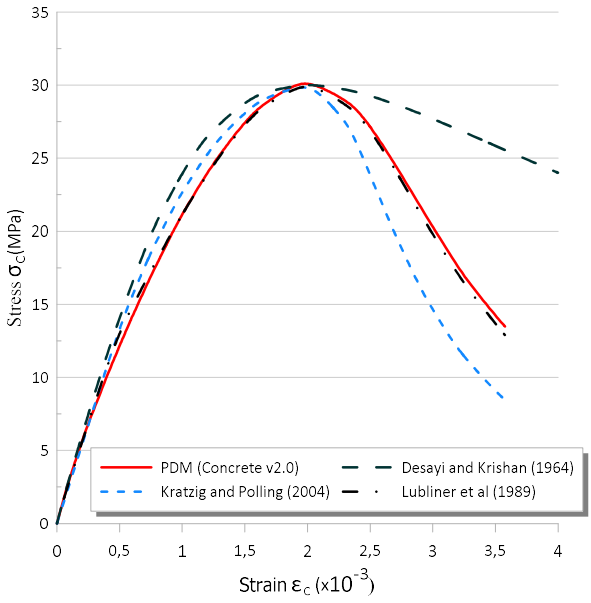


**Figure 4.11:** Compressive stress-strain curve for $f_{cm}$ = 30 MPa

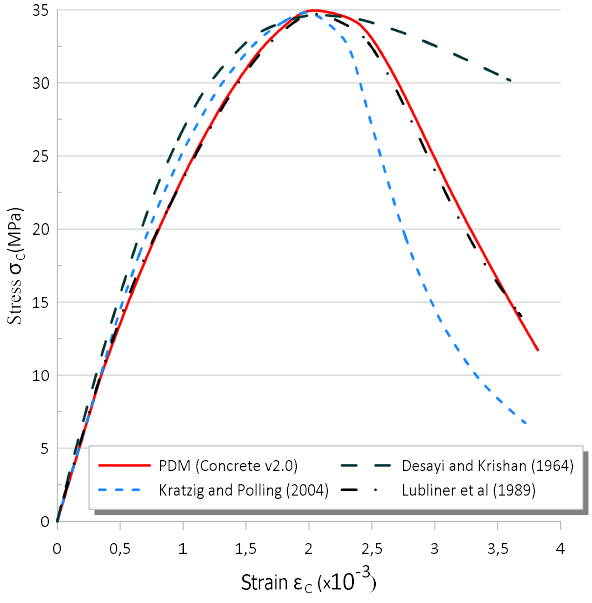


**Figure 4.12:** Compressive stress-strain curve for $f_{cm}$ = 35 MPa

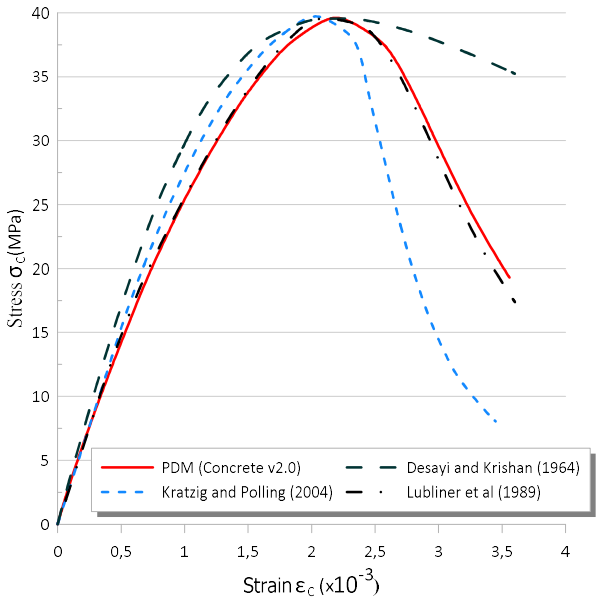


**Figure 4.13:** Compressive stress-strain curve for $f_{cm}$ = 40 MPa

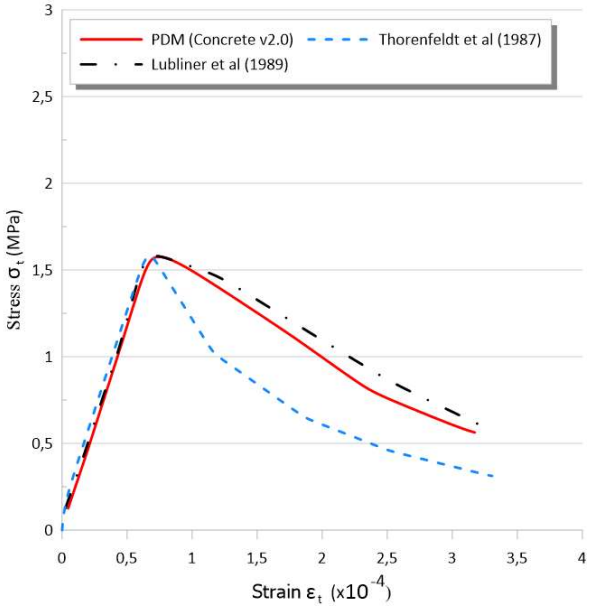


**Figure 4.14:** Tensile stress-strain curve for $f_{cm}$ = 20 MPa ($f_{tm}$ = 1.58 MPa)

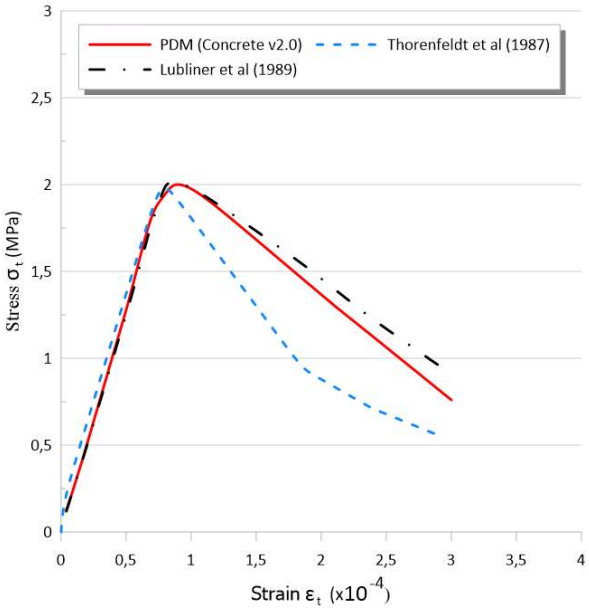


**Figure 4.15:** Tensile stress-strain curve for $f_{cm}$ = 25 MPa ($f_{tm}$ = 1.99 MPa)

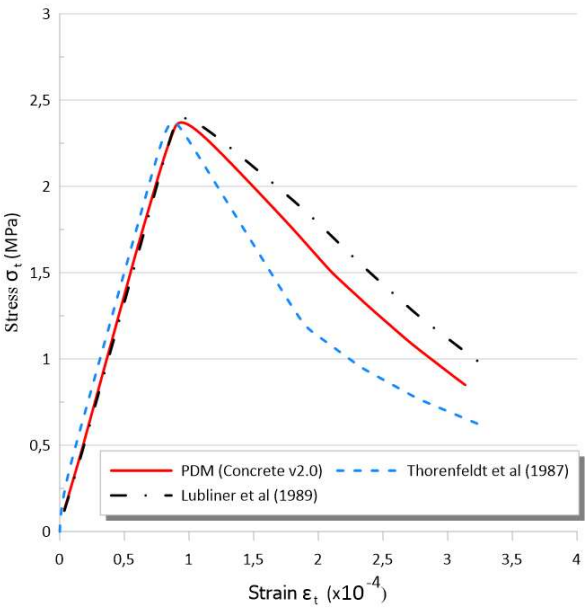


**Figure 4.16:** Tensile stress-strain curve for $f_{cm}$ = 30 MPa ($f_{tm}$ = 2.37 MPa)

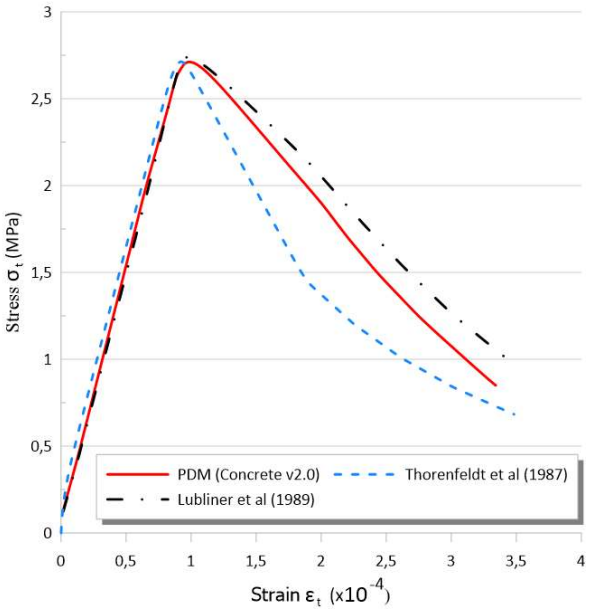


**Figure 4.17:** Tensile stress-strain curve for $f_{cm}$ = 35 MPa ($f_{tm}$ = 2.71 MPa)

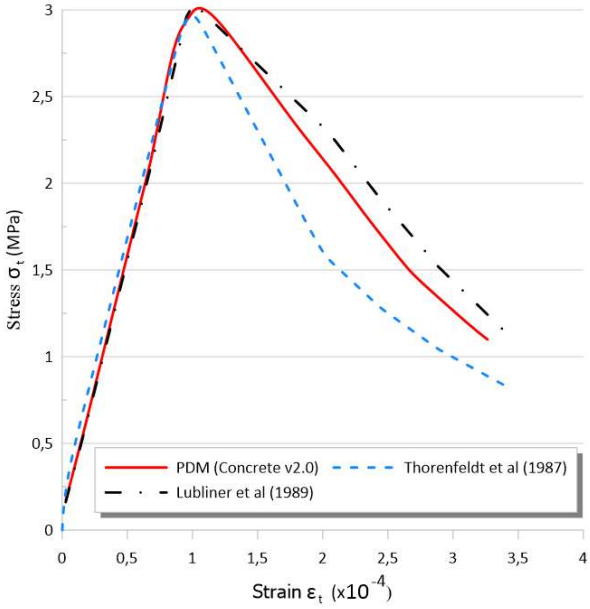


**Figure 4.18:** Tensile stress-strain curve for $f_{cm}$ = 40 MPa ($f_{tm}$ = 3.04 MPa)

## IV.4 Mesh sensitivity

To examine the mesh sensitivity, the outcomes of our computer code (for a cubic sample with the following dimensions: 250 mm long, 250 mm wide, and 250 mm high) were compared with experimental results and with closed-form solutions from the literature. Three cases of the mesh densities were considered (27, 64, and 125 elements). Table 4.3 summarizes the input data used by "Concrete v2.0.0" in the examination process for the tensile and the compressive cases. The uniaxial tensile stress at failure $\sigma_{t0}$ and the initial undamaged stiffness $E_0$ are auto-estimated according to the Model Code Recommendations [33] based on the value of the compressive strength of concrete.

In the tension case, the outcomes of "Concrete v2.0.0" were compared with the experimental results of Ahmed et al [32] and with the outcomes of the Thorenfeldt approach [102] (Figures 4.19,4.20). Two conclusions can be drawn from Figures 4.19 and 4.20. The first one is that the influence of the mesh density is very limited. The mesh insensitivity is mainly due to the use of Bakhti Approach [104] for computing the stress-strain diagrams and the damage parameters evolution in our code. The second one is that the stress-strain curves generated by our computer code are in the range of the experimental data and come very close to Thorenfeldt approach

For the compression case, the outcomes of "Concrete v2.0.0" were compared with the experimental data of Watanabe et al [103] and with the closed-form solution suggested by Kratzig and Polling [29] (which was also used by Alfarah et al [23], See section I.6.1.2.b )

**Table 4.3:** The input data for Concrete v2.0.

| | Tension | | Compression | | |
|---|---|---|---|---|---|
| | Example 01 | Example 02 | Example 01 | Example 02 | Example 03 |
| $f_{ck}$ | 39.2 MPa | 26.6 MPa | 22.5 MPa | 32 MPa | 17.5 MPa |
| $\sigma_{t0}$ | 3.48 MPa | 2.70 MPa | 2.40 MPa | 3.04 MPa | 2.04Mpa |
| $E_0$ | 32786 MPa | 28630 MPa | 27161 MPa | 30468 MPa | 25253 MPa |
| $\upsilon$ | 0.2 | | | | |
| $\varepsilon$ | 0.1 | | | | |
| $\psi$ | 5 degrees | | | | |
| $K_c$ | 0.667 | | | | |
| $f_{b0} / f_{c0}$ | 1.16 | | | | |

From figures 4.21, 4.22, we can see that the compressive stress-strain curves generated by our computer code are in the range of the experimental data provided by Watanabe et al [103]. Also, the influence of the mesh density is very limited due to the employment of the author's approach [104] for generating the stress-strain diagrams and the damage parameters evolution in "Concrete v2.0.0"

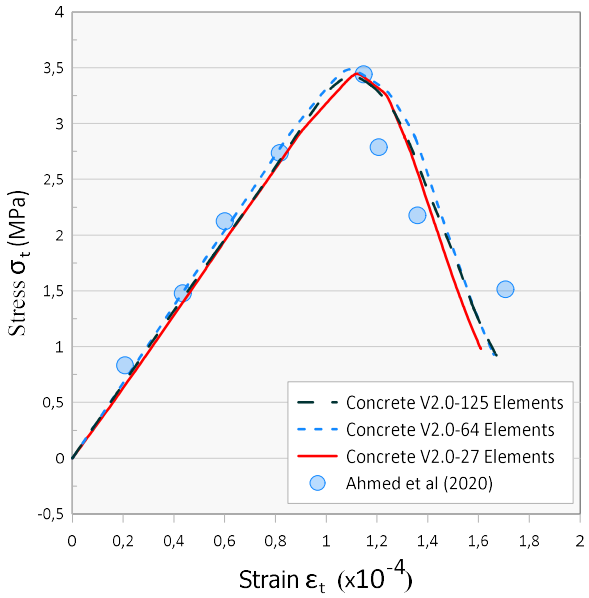


**Figure 4.19:** Tensile stress-strain curve
Mesh sensitivity-Example 01

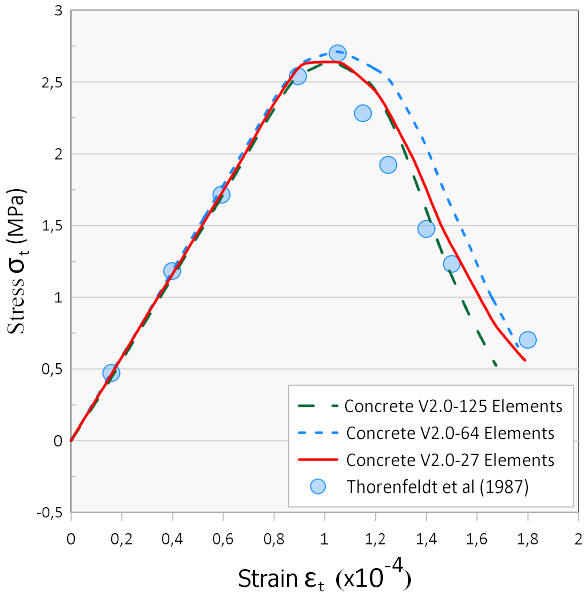


**Figure 4.20:** Tensile stress-strain curve
Mesh sensitivity - Example 02

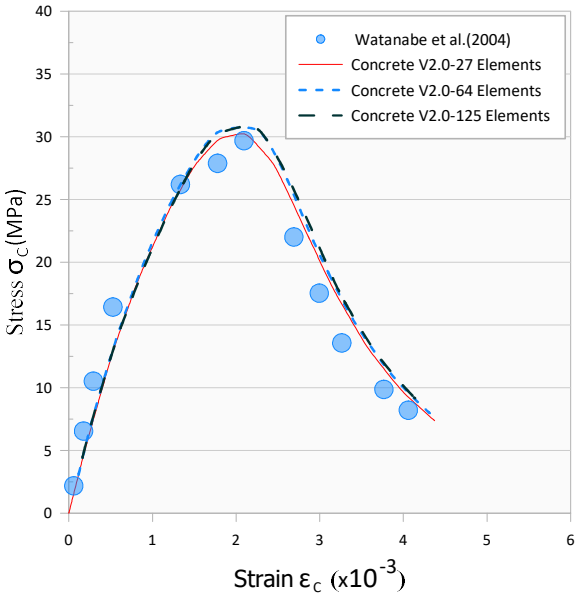


**Figure 4.21:** compressive stress-strain curve
Mesh sensitivity-Example 01

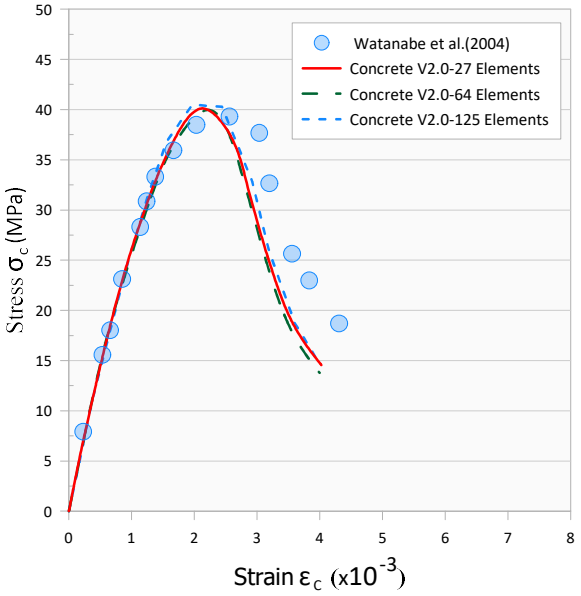


**Figure 4.22:** compressive stress-strain curve
Mesh sensitivity-Example 02

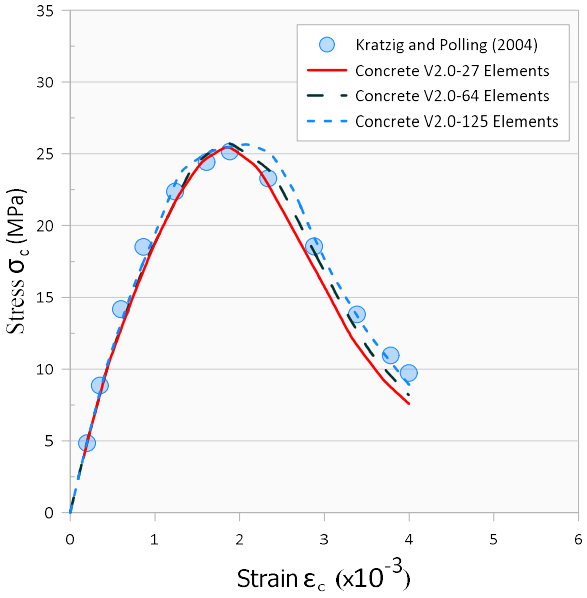


**Figure 4.23:** compressive stress-strain curve
Mesh sensitivity–Example 03

Figure 4.23 indicates that the compressive stress-strain curves generated by "Concrete v2.0.0" come very close to the outcomes generated according to Kratzig and Polling [29] approach (almost identical). Also, the influence of the mesh density is very limited due to the same reason which is the use of our approach [104] to compute the stress-strain diagrams and the damage parameters evolution in "Concrete v2.0.0"

## IV.5 Conclusion

In this chapter, a full demonstration of the efficiency of our computer code was provided. Multiple comparisons with closed-form solutions and experimental tests were carried out in order to examine the efficiency of our numerical approach for computing the stress-strain diagrams and the damage parameters evolution. Several advantages of the proposed approach can be outlined as:

- Only, the concrete compressive strength value is needed to evaluate the stress-strain and the damage parameters curves,
- Intriguingly, the stress-strain curves and the damage parameters evolution are independent of the mesh size effect ($L_{eq}$),
- Achieving a high level of accuracy in the estimation of damage parameters evolution, both in tension and compression,
- Reducing the number of parameters needed to be calibrated according to the experimental tests,
- The use of the Model Code recommendations and expandability to support other recommendations,
- The developed method is quite appropriate to incorporate into other numerical codes.

Furthermore, a comparative study of the stress-strain curves generated by “Concrete v2.0.0” and five stress-strain correlations was provided. For the compression case, the outcomes of “Concrete v2.0.0” were compared with the stress-strain curves of Desayi, Kratzig, and Lubliner. The following conclusions can be outlined:

- All curves are quite close to each other in the ascending part,
- The curves of “Concrete v2.0.0” are completely in harmony with the stress-strain curve generated according to Lubliner formula,
- For the concrete strengths less than 25 MPa, The curves of “Concrete v2.0.0” are partially in harmony with the stress-strain curve generated according to Kratzig formula. For values more than 25MPa, the curves of “Concrete v2.0.0” move away from Kratzig curve depending on the concrete strengths, especially in the descendant part,
- In the descendant part, the difference between the curves of “Concrete v2.0.0” curves and the curves of Desayi is significant.

For the tensile case the following conclusions can be made:

- All curves are quite close to each other in the ascending part,
- The outcomes of “Concrete v2.0.0” are in harmony with the stress-strain curve generated according to Lubliner formula,
- “Concrete v2.0.0” gives results partially similar to Thorenfeldt formula, especially in the descendant part.

The mesh sensitivity was exanimated in this chapter where it is demonstrated that the outcomes of “Concrete v2.0.0” are unconnected to the mesh density for both cases compression and tension.

# Conclusion

# Conclusion

Modeling the real behavior of concrete using the finite element method significantly helps the scientific community to reduce the number of required experimental tests by simulating them numerically which considerably reduces the study time and improves the profitability of research teams. The Damage Plastic Model was selected as a constitutive model to build our computer App with a view to simulate the real behavior of cylindrical and cubical concrete samples. This choice is based on several logical reasons, which are; the ability to address small and large strain, the capability to address the plasticity of concrete material, and the ability to handle the elastic stiffness degradation induced by the plastic straining in addition to the stiffness recovery. The use of the second form of the PDM requires multiple parameters namely; the stress-inelastic strain diagrams for compression and tension cases, the damage parameters evolution for compression and tension cases, the ratio of the second stress invariants on tensile and compressive meridians, the eccentricity, the ratio of biaxial compressive yield stress to uniaxial compressive yield stress, and the dilation angle. The numerical values of these parameters must reflect the real state of concrete material which required the calibration process with experimental tests for each one of them. To overcome the calibration process, default values are suggested in this work, in addition to a new numerical methodology for computing the stress-strain diagrams and the damage parameters evolution for compression and tension cases were suggested based on Lubliner and Alfarah formulas. The stress-strain curves and the damage parameters evolution in tension and compression states were calculated in accordance with the Model Code recommendations. The main advantage of this methodology is that the use of the Damage Plastic Model is no longer related to the calibration process with experimental evidence. In fact, the only parameter needed for evaluating the stress-strain and the damage parameters diagrams using the developed approach is the concrete compressive strength value. Multiple comparisons with closed-form solutions and experimental tests were carried out in order to examine the efficiency of our numerical approach. The validation process of the proposed approach proved that the outcomes for both cases; compression and tension are quite close to experimental curves and analytical solutions, in addition to the no effect of the mesh size on the outcomes of the proposed approach

Also, the full procedure of the finite element implementation of the PDM was delivered where the plastic strain estimation process was described, in addition to the closed-form of the plastic multiplier, the derivative of the yield function with respect of stresses, the derivative of the yield function with respect of the inelastic compression strain; and the derivative of the potential function with respect of stresses.

A comparative study of the stress-strain curves generated by “Concrete v2.0.0” and multiple stress-strain correlations was provided. The mesh sensitivity was exanimated in this work by a comparative study of the stress-strain curves for multiple mesh density. Several advantages of the proposed work can be outlined as:

- Only, the concrete compressive strength value is required to model the concrete behavior,
- The stress-strain curves and the damage parameters evolution are independent of the mesh size effect,
- The use of the Model Code recommendations for computing the input data (stresses and the damage parameters) and expandability to support other recommendations,
- The developed method is quite appropriate to incorporate into other numerical codes.
- The curves of “Concrete v2.0.0” are completely in harmony with the stress-strain curve generated according to several solutions from the literature,
- The ability to use another hardening function from the literature.

The developed software “Concrete v2.0.0” can be perfectly used to simulate the behavior of concrete material including the degradation process of concrete. It can be used to model cubical and cylindrical samples following the Model Code recommendation, the software can be updated to support more stress-strain correlation by changing the hardening function and re-computing, the derivative of the yield function with respect of stresses, the derivative of the yield function with respect of the inelastic compression strain. Also, the author recommends extending the software for composite material in order to model the behavior of confined concrete by composite materials.

# References


[1] J. Winskell, H. Reese, S. Dayem, et R. McCaffrey, « Global concrete report 2021 », *Pro Global Media Ltd*, p. 35, 2021.

[2] P. Edwards, « Top 10 cement producer profiles », *Global Cement Magazine*, août 2018.

[3] D.J. Chu Carreira et H. Kuang, « Stress-strain relationship for reinforced concrete in tension », *ACI journal*, vol. 84, p. 21-28, 1986.

[4] J. B. Mander, M. J. N. Priestley, et R. Park, « Observed stress-strain model for confined concrete », *Journal of Structural Engineering*, vol. 114, nº 08, p. 1827-1849, 1988.

[5] L.P. Sanez, « Discussion of 'Equation for the stress-strain curve of concrete' by Desayi and Krishnan », *ACI journal proceedings*, vol. 61, p. 1229-1235, 1964.

[6] W.F. Chen, *Constitutive Equations for Engineering Materials Vol. 1: Elasticity and Modelling*. Elsevier Publications, 1994.

[7] M.Y.H Bangash, *Concrete and Concrete Structures*. Elsevier Publications, 1989.

[8] S.H. Ahmad et S.P. Shah, « Complete triaxial stress-strain curves for concrete », *Journal of the Structural Division*, vol. 108, p. 728-742, 1982.

[9] B. Bresler et K. S. Pister, « Strength of Concrete Under Combined Stresses », *Journal Proceedings*, vol. 55, p. 321-345, 1958.

[10] M.D. Coon et R.J. Evans, « Incremental constitutive laws and their associated failure criteria with application of plain concrete », *International journal of solids structures*, vol. 8, p. 1169-1183, 1972.

[11] Z.P. Bazant et S.S. Kim, « Nonlinear creep of concrete adaptation and flow », *Journal of the Engineering Mechanics Division*, vol. 105, p. 429-446, 1979.

[12] Z.Y. Zhao et L.Q. Ren, « Failure Criterion of Concrete under Triaxial Stresses Using Neural networks », *Computer-Aided Civil and Infrastructure Engineering*, vol. 17, p. 68-73, 2002.

[13] H.D. Kang et K.J. Willam, « Localization characteristics of triaxial concrete model », *Journal of Engineering Mechanics*, vol. 125, p. 941-950, 1999.

[14] I. Imran et S.J. Pantazopoulou, « Plasticity model for concrete under triaxial compression », *Journal of Engineering Mechanics*, vol. 127, p. 281-290, 2001.

[15] P. A. Vermeer, *Non-Associated Plasticity for Soils, Concrete and Rock*, vol. 29. Delft University of Technology, 1984.

[16] P. Grassl, K. Lundgren, et K. Gylltoft, « Concrete in compression: A plasticity theory with novel hardening law », *International Journal of Solids and Structures*, vol. 39, p. 5205-5223, 2002.

[17] P.H Feenstra et R de Borst, « A composite plasticity model for concrete », *International Journal of Solids and Structures*, vol. 33, p. 707-730, 1996.

[18] A. Hillerborg, M. Modéer, et P.E. Petersson, « Analysis of crack formation and crack growth in concrete by means of fracture mechanics and finite elements », *Cement and Concrete Research*, vol. 6, p. 773-781, 1976.

[19] Z.P. Bazant, « Concrete fracture models: testing and practice », *Engineering Fracture Mechanics*, p. 165-205, 2002.

[20] Z.P. Bazant, « Endochronic inelasticity and incremental plasticity », *International Journal of Solids and Structures*, vol. 14, p. 691-714, 1978.

[21] Z.P. Bazant et H. Shieh, « Endochronic model for non-linear triaxial behaviour of concrete », *Nuclear Engineering and Design*, vol. 47, p. 305-315, 1978.

[22] Z.P. Bazant et P.D. Bhat, « Endochronic theory of inelasticity and failure of concrete », *Journal of the Engineering Mechanics*, vol. 102, p. 701-722, 1976.

[23] B. Alfarah, F. López-Almansa, et S. Oller, « New methodology for calculating damage variables evolution in Plastic Damage Model for RC structures », *Engineering Structures*, vol. 132, p. 70-86, 2017.

[24] B. Ayhan, P. Jehel, D. Brancherie, et A. Ibrahimbegovic, « Coupled damage–plasticity model for cyclic loading: Theoretical formulation and numerical implementation », *Engineering Structures*, vol. 50, p. 30-42, 2013.
[25] C. Farahmandpour, S. Dartois, M. Quiertant, Y. Berthaud, et H. Dumontet, « A concrete damage-plasticity model for FRP confined columns », *Materials and Structures*, vol. 50, nº 156, 2017.
[26] J. Lubliner, J. Oliver, S. Oller, et E. Onate, « A plastic-damage model for concrete », *Int. J. Solids Structures*, vol. 25, nº 3, p. 299-326, 1989.
[27] J. Lee et G. L. Fenves, « A plastic-damage concrete model for earthquake analysis of dams », *Earthquake engineering and structural dynamics*, vol. 27, p. 937-956, 1998.
[28] J. Lee et G. L. Fenves, « Plastic-damage model for cyclic loading of concrete structures », *J. Engrg. Mech., ASCE*, vol. 124, nº 8, p. 892-900, 1998.
[29] W. B. Kratzig et R. Polling, « An elasto-plastic damage model for reinforced concrete with minimum number of material parameters », *Computers & Structures*, vol. 82, p. 1201-1215, 2004.
[30] S. Oller, E. Onate, J. Oliver, et J. Lubliner, « Finite element nonlinear analysis of concrete structures using a "plastic-damage model" », *Engineering Fracture Mechanics*, vol. 35, nº 1/2/3, p. 219-231, 1990.
[31] J. Lee et G. L. Fenves, « A return-mapping algorithm for plastic-damage models: 3-D and plane stress formulation », *Int. J. Numer. Meth. Engng*, vol. 50, p. 487-506, 2001.
[32] B. Ahmed, G.Z. Voyiadjis, et T. Park, « Damaged Plasticity Model for Concrete Using Scalar Damage Variables with a Novel Stress Decomposition », vol. 191–192, p. 56-75, 2020.
[33] CEB-FIP, *Model Code 2010*. London: Thomas Telford, 2010.
[34] R. R. Babu, G.S. Benipal, et A.K. Singh, « Constitutive modeling of concrete: An overview », *Asian Journal of Civil Engineering*, vol. 6, nº 4, p. 211-246, 2005.
[35] M.R. Javanmardi et M.R. Maheri, « Extended finite element method and anisotropic damage plasticity for modelling crack propagation in concrete », *Finite Elements in Analysis and Design*, vol. 165, p. 1-20, 2019.
[36] M. Poliotti et J.M. Bairán, « A new concrete plastic-damage model with an evolutive dilatancy parameter », *Engineering Structures*, vol. 189, p. 541-549, 2019.
[37] M.A.L. Silva, J.C.P.H. Gamage, et S. Fawzia, « Performance of slab-column connections of flat slabs strengthened with carbon fiber reinforced polymers », *Case Studies in Construction Materials*, vol. 11, p. e00275, 2019.
[38] W. Ren, L. H. Sneed, Y. Yang, et R. He, « Numerical Simulation of Prestressed Precast Concrete Bridge Deck Panels Using Damage Plasticity Model », *International Journal of Concrete Structures and Materials*, vol. 9, nº 1, p. 45-54, 2015.
[39] H. Othman et H. Marzouk, « A study on damage mechanism modelling of shield tunnel under unloading based on damage–plasticity model of concrete », *International Journal of Impact Engineering*, vol. 114, p. 20-31, 2018.
[40] P.H. Menetrey et K.J.Willam, « Triaxial failure criterion for concrete and its generalization », *ACI. Struct. J*, vol. 92, p. 1995, 318 311apr. J.-C..
[41] N.S. Ottosen, « A failure criterion for concrete », *J. Engrg. Mech., ASCE*, vol. 103, p. 1977, 535 527.
[42] D.J. Han et W.F.Chen, « strain space plasticity formulation for hardening-softening materials with elastoplastic coupling », *Int. J. Solids Struct*, vol. 22, p. 935-950, 1986.
[43] E.N. Dvorkin, A.M. Cuitino, et G. Gioia, « A concrete material model based on nonassociated plasticity and fracture », *Engrg. Comput*, vol. 6, p. 281-294, 1989.
[44] P.A. Vermeer et R.de Borst, « Non-associated plasticity for soils, concrete and rock », *Heron*, vol. 29, nº 3, p. 3-64, 1984.

[45] K.C. Valanis, « A theory of visco-plasticity with out a yield surface, part I: General theory », *Archives of Mech*, vol. 23, p. 517-551, 1971.
[46] D.V. Reddy et K.R. Gopal, « Endochronic constitutive modeling of marine fiber reinforced concrete », *Comp. Modeling of RC struct*, p. 154-186, 1986.
[47] P. Desayi et S. Krishnan, « Equation for the stress-strain curve of concrete », *ACI journal*, vol. 61, p. 345-350, 1964.
[48] G.M. Smith et L.E. Young, « Ultimate flexural analysis based on stress-strain curves of cylinders », *ACI journal*, vol. 53, p. 597-610, 1956.
[49] R.M. Richard et B.J. Abbott, « Versatile elastic-plastic stress-strain formula », *J. Engrg. Mech, ASCE*, vol. 101, p. 511-515, 1975.
[50] A. Mohamad Ali, B.j. Farid, et A.I.M. Al-Janabi, « Stress-Strain Relationship for Concrete in Compression Madel of Local Materials », *JKAU: Eng. Sci*, vol. 2, p. 183-194, 1990.
[51] D.J. Carreira et K.H. Chu, « Stress-Strain Relationship for Plain Concrete in Compression », *ACI journal*, vol. 82, n° 6, p. 797-804, 1985.
[52] D.J. Carreira et K.H. Chu, « Stress-strain relationship for reinforced concrete in tension », *ACI journal*, vol. 84, p. 21-28, 1986.
[53] J.W. Dougill, « Some remarks on path independence in the small in plasticity », *Quart.App. Math*, vol. 32, p. 233-243, 1975.
[54] J.W. Dougill, « On sTable progressively fracturing solids », *ZAMP*, vol. 27, p. 423-437, 1976.
[55] J.W. Ju, « On energy based coupled elasto plastic damage theories: Constitutive modelling and computational aspects », *Int. J. Solids and Struct*, vol. 25, p. 803-833, 1989.
[56] G.Z. Voyiadjis et J.M. Abu Lebdeh, « Damage model for concrete using the bounding surface concept », *J. Engrg. Mech., ASCE*, vol. 119, p. 1865-1885, 1993.
[57] D. Krajcinovic, « Damage mechanics », *Mech. Mat*, vol. 8, p. 117-197.
[58] A.K. Singh, « Finite element analysis of damage coupled elastoplastic problems based on continuum damage mechanics ». Ph.D thesis, Indian Institute of Science, Bangalore, India, 1999.
[59] L.M. Kachanov, « Introduction to Continuum Damage Mechanics ». Kulwer Academic Publishers, Dordrecht, 1986.
[60] Yu. N. Rabotnov, « Creep Problems in Structural Members ». North Holland Publishing Company, Amsterdam, 1969.
[61] J.C. Simo et J.W. Ju, « Strain and stress based continuum damage models-I.Formulation », *Int. J. Solids. Struct*, vol. 23, p. 821-840, 1987.
[62] J.C. Simo et J.W. Ju, « Strain and stress based continuum damage models-II.Computational aspects », *Int. J. Solids. Struct*, vol. 23, p. 841-869, 1987.
[63] J.W. Ju, « Isotropic and anisotropic damage variables in continuum damage mechanics », *J.Engrg. Mech., ASCE*, vol. 116, p. 2764-2770, 1990.
[64] J. Lamaitre, « How to use damage mechanics », *Nucl. Engrg. Design*, vol. 80, p. 233-245, 1984.
[65] J. Lamaitre, « A continuous damage mechanics model for ductile fracture », *J. Engrg. Mat. Tech*, vol. 107, p. 83-89, 1985.
[66] J. Lamaitre, « A Course on Damage Mechanics ». Springer-Verlag, 1992.
[67] J.L. Chaboche, « Continuum damage mechanics I. General concepts », *J. App. Mech., ASME*, vol. 55, p. 55-59, 1988.
[68] J.L. Chaboche, « Continuum damage mechanics II. Damage growth, crack initiation and crack growth », *J. App. Mech., ASME*, vol. 55, p. 65-72, 1988.
[69] D. Krajcinovic et G.U. Foneska, « The continuous damage theory of brittle materials Part I: General theory », *J. App. Mech*, vol. 48, p. 809-815, 1981.
[70] D. Krajcinovic, « Constitutive equations for damaging materials », *J. Appl. Mech*, vol. 50, p. 355-360, 1983.

[71] J.L. Chaboche, « Mechanical Behaviour of Anisotropic Solid ». Ed. J.P. Boehler, Martinus Nijhoff, 1982.
[72] M. Ortiz, « A constitutive theory for inelastic behaviour of concrete », *Mech. Mater*, vol. 4, p. 67-93, 1985.
[73] D.W. Nicholson, « Constitutive model for rapidely damaged structural material », *Acta. Mech*, vol. 39, p. 195-205, 1981.
[74] J. Y. Wu, J. Li, et R. Faria, « An energy release rate-based plastic-damage model for concrete », *International Journal of Solids and Structures*, vol. 43, nº 3-4, p. 583-612, 2006.
[75] Hibbitt, Karlsson, et Sorensen, *Abaqus analysis user's manual Volume III: Materials version 6.10*. Dassault Systémes, 2010.
[76] J. Zhang, Z. Zhang, et C. Chen, « Yield criterion in plastic-damage models for concrete », *Acta Mechanica Solida Sinica*, vol. 23, nº 3, 2010.
[77] M. Hafezolghorani, F. Hejazi, R. Vaghei, M.S. Bin Jaafar, et K. Karimzade, « Simplified Damage Plasticity Model for Concrete », *Structural Engineering International*, nº 1, p. 68-78, 2017.
[78] T. Yu, J.G. Teng, Y.L. Wong, et S.L. Dong, « Finite element modeling of confined concrete-II: Plastic-damage model », *Engineering Structures*, vol. 32, p. 680-691, 2010.
[79] W. Demin et H. Fukang, « Investigation for plastic damage constitutive models of the concrete material », *6th International Workshop on Performance, Protection & Strengthening of Structures under Extreme Loading (PROTECT2017), Guangzhou (Canton), China*, 2017.
[80] D.A Hordijk, « Tensile and tensile fatigue behavior of concrete; experiments, modeling and analyses », *Heron*, vol. 37, nº 1, p. 9-79, 1992.
[81] M. Szczecina et A. Winnicki, « Calibration of the CDP model parameters in Abaqus », *The 2015 World Congress on Advances in Structural Engineering and Mechanics (ASEM15),Incheon, Korea*, 2015.
[82] M. Szczecina et A. Winnicki, « Numerical simulations of corners in RC frames using Strut-and-Tie method and CDP model », *XIII International Conference on Computational Plasticity. Fundamentals and Applications (COMPLAS XIII), Barcelona, Spain*, 2015.
[83] Y. Sümer et M. Aktaş, « Defining parameters for concrete damage plasticity model », *Challenge Journal of Structural Mechanics*, vol. 1, nº 2, p. 149-155, 2015.
[84] A. Demir, H. Ozturk, K. Edip, M. Stojmanovska, et A. Bogdanovic, « Effect of viscosity parameter on the numerical simulation of reinforced concrete deep beam behavior », *The Online Journal of Science and Technology*, vol. 8, nº 3, p. 50-56, 2018.
[85] R. Bhartiya, D.R.Sahoo, et A.Verma, « Modified damaged plasticity and variable confinement modelling of rectangular CFT columns », *Journal of Constructional Steel Research*, vol. 176, p. 106426, 2021.
[86] L.M. e Silva, A.L. Christoforo, et R.C. Carvalho, « Calibration of Concrete Damaged Plasticity Model parameters for shear walls. », *Revista Matéria*, vol. 26, nº 1, 2021.
[87] H. Behnam, J.S. Kuang, et B. Samali, « Parametric finite element analysis of RC wide beam-column connections », *Computers and Structures*, vol. 205, p. 28-44, 2018.
[88] X. Yangjian, Z. Jianting, L. Yanling, et X. Runchuan, « Concrete plastic-damage factor for finite element analysis: Concept, simulation, and experiment », *Advances in Mechanical Engineering*, vol. 9, nº 9, p. 1-10, 2017.
[89] D.M.Potts et L. Zdravkovic, *Finite element analysis in geotechnical engineering: Theory*, Thomas Telford. 1999.
[90] E.A. de Souza Neto, D. Peric, et D. R. J. Owen, *Computational Methods for Plasticity: Theory and Applications*. John Wiley & Sons, United Kingdom, 2008.
[91] C. Woltering, « Triangle.NET », *Github*. https://github.com/wo80/Triangle.NET
[92] « OpenTK ». https://opentk.net/
[93] « Ribbon WinForms », *Github*. https://github.com/RibbonWinForms/RibbonWinForms

[94] « Math.NET Numerics ». https://numerics.mathdotnet.com/
[95] O.J. Dahl et K. Nygaard, « SIMULA an ALGOL-Based Simulation Language », *Communications of the ACM*, vol. 9, nº 9, p. 671-678, 1966.
[96] B. W. R. Forde, R. O. Foschi, et S. F. Stiemer, « Object-oriented finite element analysis », *Computers & Structures*, vol. 34, p. 355-374, 1990.
[97] R.I Mackie, « Object oriented programming of the finite element method », *International Journal for Numerical Methods In Engineering*, vol. 35, p. 425-436, 1992.
[98] Y. Dubois-Pelerin, T. Zimmermann, et P. Bomme, « Object-oriented finite element programming: II. A prototype program in smalltalk », *Computer Methods in Applied Mechanics and Engineering*, vol. 98, p. 361-397, 1992.
[99] S. Kumar, « Object-Oriented Finite Element Programming for Engineering Analysis in C++ », *Journal of Software*, vol. 5, nº 7, p. 689-696, 2010.
[100] P. D. Alves, F. B. Barros, et R. L.S. Pitangueira, « An object-oriented approach to the Generalized Finite Element Method », *Advances in Engineering Software*, vol. 59, p. 1-18, 2013.
[101] R. Benjamin *et al.*, « CastLab: an object-oriented finite element toolbox within the Matlab environment for educational and research purposes in computational solid mechanics », *Advances in Engineering Software*, vol. 128, p. 136-151, 2019.
[102] E. Thorenfeldt, A. Tomaszewicz, et J. J. Jensen, « Mechanical properties of high-strength concrete and application in design », *in Proc. Symposium on Utilization of High-Strength Concrete, Tapir, Trondheim, Norway*, p. 149-159, 1987.
[103] K. Watanabe, J. Niwa, H. Yokota, et M. Iwanami, « Experimental Study on Stress-Strain Curve of Concrete Considering Localized Failure in Compression », *Journal of Advanced Concrete Technology*, vol. 2, nº 3, p. 395-407, 2004.
[104] R. Bakhti, B.Benahmed, A. Laib, et M.T. Alfach., « New approach for computing damage parameters evolution in Plastic Damage Model for concrete », *Case Studies in Construction Materials*, vol. 16, nº e00834, p. 1-19, 2022.

# Appendix

The source code of Concrete v2.0.0 is available under MIT licensing on GitHub through the link:

https://github.com/BakhtiSoftwares/Concrete

# Mesh subroutines

The GenerateMeshCylindre subroutine is written as:

```
Private Sub GenerateMeshCylindre(Diametre As Double, Haut As Double, E As Double, V As Double, Fb0_Fc0
                As Double, Kc As Double, PsiDegre As Double,Fck As Double, Excent As Double, Rhou As
                Double, Comprission As List(Of DoublePoint), Tension As List(Of DoublePoint))

Dim Counteur As List(Of DoublePoint) = CalculerPiremetreCylindre(Diametre)
        Dim OurMesh As Mesh = MeshGenerator(Counteur)
        QuadMesh = QMesh(New QuadMesh(OurMesh))
        Delete double QUAD
        Dim Termine As Boolean = False
        Do Until Termine
            Dim Count As Integer = 0
            Dim ExistQuad As Boolean = False
            For i = 0 To QuadMesh.Count - 1
                For j = i + 1 To QuadMesh.Count - 1
                    If SameQuad(QuadMesh.Item(i), QuadMesh.Item(j)) Then
                        ExistQuad = True
                        Count = i
                        Exit For
                    End If

                Next
                If ExistQuad Then Exit For
            Next
            If ExistQuad Then
                QuadMesh.Remove(QuadMesh.Item(Count))
            Else
                Termine = True
            End If
        Loop
        Noeuds.Clear()
        Dim NumbrElemet As Integer = Haut / MaxLong
        Dim Cont As Integer = -1
        Dim LongReal As Double = Haut / NumbrElemet
        NbrNoeudEtage = ListOfVertex.Count
        Dim Dis As Double = 999999999
        Dim Dis1 As Double
        For i = 0 To ListOfVertex.Count - 1
            Dis1 = Sqrt(ListOfVertex.Item(i).x ^ 2 + ListOfVertex.Item(i).y ^ 2)
            If Dis1 < Dis Then
                NoeudProcheCentre.x = ListOfVertex.Item(i).x
                NoeudProcheCentre.y = ListOfVertex.Item(i).y
            End If
        Next
        Dim Z As Double
        Do Until Z > Haut
            For i = 0 To ListOfVertex.Count - 1
                Cont = Cont + 1
                Dim NewNoeud As New Node
                NewNoeud.Ident = Cont
                NewNoeud.Coord(1) = ListOfVertex.Item(i).x
                NewNoeud.Coord(2) = ListOfVertex.Item(i).y
                NewNoeud.Coord(3) = Z
                Noeuds.Add(NewNoeud)
            Next
            Z = Z + LongReal
            If Abs(Z - Haut) < 0.0001 Then Z = Haut
        Loop
        Z = LongReal
        Elements.Clear()
        Cont = -1
        Dim Etage, NumberNoued As Integer
        Do Until Z > Haut
            For i = 0 To QuadMesh.Count - 1
                Cont = Cont + 1
                Dim ListNoeud As New List(Of Node)
                NumberNoued = QuadMesh.Item(i).S1 + Etage * ListOfVertex.Count
                ListNoeud.Add(Noeuds.Item(NumberNoued))
                NumberNoued = QuadMesh.Item(i).S2 + Etage * ListOfVertex.Count
```

```
                ListNoeud.Add(Noeuds.Item(NumberNoued))
                NumberNoued = QuadMesh.Item(i).S3 + Etage * ListOfVertex.Count
                ListNoeud.Add(Noeuds.Item(NumberNoued))
                NumberNoued = QuadMesh.Item(i).S4 + Etage * ListOfVertex.Count
                ListNoeud.Add(Noeuds.Item(NumberNoued))
                NumberNoued = QuadMesh.Item(i).S1 + (Etage + 1) * ListOfVertex.Count
                ListNoeud.Add(Noeuds.Item(NumberNoued))
                NumberNoued = QuadMesh.Item(i).S2 + (Etage + 1) * ListOfVertex.Count
                ListNoeud.Add(Noeuds.Item(NumberNoued))
                NumberNoued = QuadMesh.Item(i).S3 + (Etage + 1) * ListOfVertex.Count
                ListNoeud.Add(Noeuds.Item(NumberNoued))
                NumberNoued = QuadMesh.Item(i).S4 + (Etage + 1) * ListOfVertex.Count
                ListNoeud.Add(Noeuds.Item(NumberNoued))
                Elements.Add(New BrickEightNodes(Cont, E, V, Fb0_Fc0, Kc, PsiDegre, Fck, Excent, Rhou,
Comprission, Tension, ListNoeud))
            Next
            Z = Z + LongReal
            If Abs(Z - Haut) < 0.0001 Then Z = Haut
            Etage = Etage + 1
        Loop
End Sub
```

With MeshGenerator is the subroutine that generates 2D mesh of the cylinder section using Triangle.Net library

```
Private Function MeshGenerator(Polygon As List(Of DoublePoint)) As Mesh
        Dim Resulat As Mesh
        Dim MyPolygon As New Polygon
        Dim VertexCount As New List(Of Vertex)
        For i = 0 To Polygon.Count - 1
            MyPolygon.Add(New Vertex(Polygon.Item(i).x, Polygon.Item(i).y))
        Next
        Dim Options As New ConstraintOptions
        Dim Quality As New QualityOptions()
        Quality.MaximumAngle = 130
        Quality.MinimumAngle = 25
        Quality.MaximumArea = MaxLong * MaxLong
        Resulat = MyPolygon.Triangulate(Options, Quality)
        Return Resulat
End Function
```

The GenerateMeshCube subroutine is written as;

```
Private Sub GenerateMeshCube(ZHaut As Double, Ylar As Double, XLon As Double, E As Double, V As
                Double, Fb0_Fc0 As Double, Kc As Double, PsiDegre As Double,Fck As Double, Excent As
                Double, Rhou As Double, Comprission As List(Of DoublePoint), Tension As List(Of
                DoublePoint))
        Dim LineNumber As Integer = 0
        ListOfVertex.Clear()
        ListOfVertex = CalculatePointsCube(Ylar, XLon, LineNumber)
        Dim Dtot As Double = Ylar
        Dim DisY As Double = Ylar * MaillgeParametre / 100
        DisY = Ylar / DisY
        DisY = Ylar / (Int(DisY))
        Dim DisCour As Double = DisY
        Dim Cont As Integer = 1
        QuadMesh.Clear()
        Do
            For i = 0 To LineNumber - 2
                QuadMesh.Add(New Quadralateral With {.S1 = i + (Cont - 1) * LineNumber, .S2 = i +
               (Cont - 1) * LineNumber + 1,                                .S3 = i + Cont * LineNumber +
               1, .S4 = i + Cont * LineNumber})
            Next
            Cont = Cont + 1
            If DisCour >= Dtot Then Exit Do
            DisCour = DisCour + DisY
            If DisCour >= Dtot Then
                For i = 0 To LineNumber - 2
                    QuadMesh.Add(New Quadralateral With {.S1 = i + (Cont - 1) * LineNumber, .S2 = i +
                (Cont - 1) * LineNumber + 1,                                .S3 = i + Cont * LineNumber +
                1, .S4 = i + Cont * LineNumber})
```

```
            Next
            Exit Do
        End If
    Loop        'Delete double QUAD
    Dim Termine As Boolean = False
    Do Until Termine
        Dim Count As Integer = 0
        Dim ExistQuad As Boolean = False
        For i = 0 To QuadMesh.Count - 1
            For j = i + 1 To QuadMesh.Count - 1
                If SameQuad(QuadMesh.Item(i), QuadMesh.Item(j)) Then
                    ExistQuad = True
                    Count = i
                    Exit For
                End If
            Next
            If ExistQuad Then Exit For
        Next
        If ExistQuad Then
            QuadMesh.Remove(QuadMesh.Item(Count))
        Else
            Termine = True
        End If
    Loop
    Noeuds.Clear()
    Dim NumbrElemet As Integer = Haut / MaxLong
    Dim LongReal As Double = Haut / NumbrElemet
    NbrNoeudEtage = ListOfVertex.Count
    Dim Dis As Double = 999999999
    Dim Dis1 As Double
    For i = 0 To ListOfVertex.Count - 1
        Dis1 = Sqrt(ListOfVertex.Item(i).x ^ 2 + ListOfVertex.Item(i).y ^ 2)
        If Dis1 < Dis Then
            NoeudProcheCentre.x = ListOfVertex.Item(i).x
            NoeudProcheCentre.y = ListOfVertex.Item(i).y
        End If
    Next
    Cont = -1
    Dim Z As Double
    Do Until Z > Haut
        For i = 0 To ListOfVertex.Count - 1
            Cont = Cont + 1
            Dim NewNoeud As New Node
            NewNoeud.Ident = Cont
            NewNoeud.Coord(1) = ListOfVertex.Item(i).x
            NewNoeud.Coord(2) = ListOfVertex.Item(i).y
            NewNoeud.Coord(3) = Z
            Noeuds.Add(NewNoeud)
        Next
        Z = Z + LongReal
        If Abs(Z - Haut) < 0.0001 Then Z = Haut
    Loop
    Z = LongReal
    Cont = -1
    Elements.Clear()
    Dim Etage, NumberNoued As Integer
    Do Until Z > Haut
        For i = 0 To QuadMesh.Count - 1
            Cont = Cont + 1
            Dim ListNoeud As New List(Of Node)
            NumberNoued = QuadMesh.Item(i).S1 + Etage * ListOfVertex.Count  ' Noeud 01
            ListNoeud.Add(Noeuds.Item(NumberNoued))
            NumberNoued = QuadMesh.Item(i).S2 + Etage * ListOfVertex.Count  ' Noeud 02
            ListNoeud.Add(Noeuds.Item(NumberNoued))
            NumberNoued = QuadMesh.Item(i).S3 + Etage * ListOfVertex.Count  'Noeud 03
            ListNoeud.Add(Noeuds.Item(NumberNoued))
            NumberNoued = QuadMesh.Item(i).S4 + Etage * ListOfVertex.Count  ' Noeud 04
            ListNoeud.Add(Noeuds.Item(NumberNoued))
            NumberNoued = QuadMesh.Item(i).S1 + (Etage + 1) * ListOfVertex.Count ' Noeud 05
            ListNoeud.Add(Noeuds.Item(NumberNoued))
            NumberNoued = QuadMesh.Item(i).S2 + (Etage + 1) * ListOfVertex.Count   'Noeud 06
            ListNoeud.Add(Noeuds.Item(NumberNoued))
            NumberNoued = QuadMesh.Item(i).S3 + (Etage + 1) * ListOfVertex.Count  'Noeud 07
            ListNoeud.Add(Noeuds.Item(NumberNoued))
            NumberNoued = QuadMesh.Item(i).S4 + (Etage + 1) * ListOfVertex.Count ' Noeud 08
```

```
                ListNoeud.Add(Noeuds.Item(NumberNoued))
                Elements.Add(New BrickEightNodes(Cont, E, V, Fb0_Fc0, Kc, PsiDegre, Fck, Excent, Rhou,
Compression, Tension, ListNoeud))
            Next
            Z = Z + LongReal
            If Abs(Z - Haut) < 0.0001 Then Z = Haut
            Etage = Etage + 1
        Loop
End Sub
```

# Drawing subroutines

```
Public Sub ChargerGLControl(MyGlControl As GLControl)
        GL.ClearColor(Color.White)
        GL.Clear(ClearBufferMask.ColorBufferBit)
        GL.MatrixMode(MatrixMode.Projection)
        GL.LoadIdentity()
        GL.PolygonMode(MaterialFace.FrontAndBack, PolygonMode.Line)
        GL.Clear(ClearBufferMask.ColorBufferBit)
        GL.ClearColor(Color.Black)
        GL.MatrixMode(MatrixMode.Projection)
        GL.LoadIdentity()
        MyGlControl.SwapBuffers()
End Sub

Public Sub ReDraw2D(MyGlControl As GLControl)
        Dim w, h As Integer
        w = MyGlControl.Width
        h = MyGlControl.Height
        Dim Rapport As Double = w / h
        GL.ClearColor(Color.Black)
        GL.Clear(ClearBufferMask.ColorBufferBit)
        GL.LoadIdentity()
        GL.MatrixMode(MatrixMode.Projection)
        GL.Ortho(-1, 1, -1, 1, -1, 1)
        GL.Viewport(0, 0, w, h)
        OurProblem.Draw2D(Color.Red)
End Sub

Public Sub ReDraw3D(MyGlControl As GLControl)
        Dim w, h As Integer
        w = MyGlControl.Width
        h = MyGlControl.Height
        Dim Rapport As Double = w / h
        Dim Prespective As Matrix4 = Matrix4.CreatePerspectiveFieldOfView(1, Rapport, 0.01, 100)
        Dim LoackAt As Matrix4 = Matrix4.LookAt(eyeX, eyeY, eyeZ, targetX, targetY, targetZ, 0, 0, 1)
        GL.MatrixMode(MatrixMode.Projection)
        GL.LoadIdentity()
        GL.LoadMatrix(Prespective)
        GL.MatrixMode(MatrixMode.Modelview)
        GL.LoadIdentity()
        GL.LoadMatrix(LoackAt)
        GL.Viewport(0, 0, w, h)
        GL.ClearColor(Color.Black)
        GL.Clear(ClearBufferMask.ColorBufferBit)
        OurProblem.Draw3D(True, Color.Red)
        OurProblem.Draw3D(False, Color.SkyBlue, EchelleAffichage)
End Sub

Public Sub Draw3D(InitialShape As Boolean, PrintColor As Color, Optional Echelle As Integer = 10)
If IsNothing(QuadMesh) Then Exit Sub
        Dim ContIndex As Integer = 0
        GL.Enable(EnableCap.Light0)
        GL.Enable(EnableCap.LineSmooth)
        If InitialShape Then
        GL.PushMatrix()
            For Each Elem In Elements
                GL.PolygonMode(MaterialFace.FrontAndBack, PolygonMode.Line)
                GL.Color3(Color.BlueViolet)
                DessignerCube(Elem.ListNoeud.Item(0), Elem.ListNoeud.Item(1), Elem.ListNoeud.Item(2),
Elem.ListNoeud.Item(3),Elem.ListNoeud.Item(4), Elem.ListNoeud.Item(5), Elem.ListNoeud.Item(6),
Elem.ListNoeud.Item(7))
                GL.PolygonMode(MaterialFace.FrontAndBack, PolygonMode.Fill)
                GL.Color3(Color.Brown)
```

```
                DessignerCube(Elem.ListNoeud.Item(0), Elem.ListNoeud.Item(1), Elem.ListNoeud.Item(2),
Elem.ListNoeud.Item(3),Elem.ListNoeud.Item(4), Elem.ListNoeud.Item(5), Elem.ListNoeud.Item(6),
Elem.ListNoeud.Item(7))
            Next
            GL.PopMatrix()
        Else
            GL.PushMatrix()
            For Each Elem In Elements
                GL.LineWidth(1)
                GL.PolygonMode(MaterialFace.FrontAndBack, PolygonMode.Line)
                GL.Color3(Color.DarkBlue)
                DessignerCubeDeplacer(Elem.ListNoeud.Item(0), Elem.ListNoeud.Item(1),
Elem.ListNoeud.Item(2), Elem.ListNoeud.Item(3),Elem.ListNoeud.Item(4), Elem.ListNoeud.Item(5),
Elem.ListNoeud.Item(6), Elem.ListNoeud.Item(7), Echelle)
                GL.LineWidth(2)
                GL.PolygonMode(MaterialFace.FrontAndBack, PolygonMode.Fill)
                GL.Color3(PrintColor)
                DessignerCubeDeplacer(Elem.ListNoeud.Item(0), Elem.ListNoeud.Item(1),
Elem.ListNoeud.Item(2), Elem.ListNoeud.Item(3),Elem.ListNoeud.Item(4), Elem.ListNoeud.Item(5),
Elem.ListNoeud.Item(6), Elem.ListNoeud.Item(7), Echelle)
                GL.LineWidth(1)
            Next
            GL.PopMatrix()
        End If
End Sub

Public Sub Draw2D(PrintColor As Color)
        If IsNothing(QuadMesh) Then Exit Sub
        Dim ContIndex As Integer = 0
        GL.PushMatrix()
        GL.Color3(PrintColor)
        GL.Disable(EnableCap.Light0)
        GL.Disable(EnableCap.LineSmooth)
        GL.LineWidth(1)
        For Each Quad In QuadMesh
            GL.Begin(PrimitiveType.LineLoop)
            GL.Vertex2(ListOfVertex.Item(Quad.S1).x, ListOfVertex.Item(Quad.S1).y)
            GL.Vertex2(ListOfVertex.Item(Quad.S2).x, ListOfVertex.Item(Quad.S2).y)
            GL.Vertex2(ListOfVertex.Item(Quad.S3).x, ListOfVertex.Item(Quad.S3).y)
            GL.Vertex2(ListOfVertex.Item(Quad.S4).x, ListOfVertex.Item(Quad.S4).y)
            GL.End()
        Next
        GL.PopMatrix()
End Sub

Private Sub DessignerCube(Noued1 As Node, Noued2 As Node, Noued3 As Node, Noued4 As Node,Noued5 As
Node, Noued6 As Node, Noued7 As Node, Noued8 As Node)
        GL.Begin(PrimitiveType.Polygon)
        GL.Vertex3(Noued1.Coord(1), Noued1.Coord(2), Noued1.Coord(3))
        GL.Vertex3(Noued2.Coord(1), Noued2.Coord(2), Noued2.Coord(3))
        GL.Vertex3(Noued3.Coord(1), Noued3.Coord(2), Noued3.Coord(3))
        GL.Vertex3(Noued4.Coord(1), Noued4.Coord(2), Noued4.Coord(3))
        GL.End()

        GL.Begin(PrimitiveType.Polygon)
        GL.Vertex3(Noued5.Coord(1), Noued5.Coord(2), Noued5.Coord(3))
        GL.Vertex3(Noued6.Coord(1), Noued6.Coord(2), Noued6.Coord(3))
        GL.Vertex3(Noued7.Coord(1), Noued7.Coord(2), Noued7.Coord(3))
        GL.Vertex3(Noued8.Coord(1), Noued8.Coord(2), Noued8.Coord(3))
        GL.End()

        GL.Begin(PrimitiveType.Polygon)
        GL.Vertex3(Noued1.Coord(1), Noued1.Coord(2), Noued1.Coord(3))
        GL.Vertex3(Noued2.Coord(1), Noued2.Coord(2), Noued2.Coord(3))
        GL.Vertex3(Noued6.Coord(1), Noued6.Coord(2), Noued6.Coord(3))
        GL.Vertex3(Noued5.Coord(1), Noued5.Coord(2), Noued5.Coord(3))
        GL.End()

        GL.Begin(PrimitiveType.Polygon)
        GL.Vertex3(Noued3.Coord(1), Noued3.Coord(2), Noued3.Coord(3))
        GL.Vertex3(Noued4.Coord(1), Noued4.Coord(2), Noued4.Coord(3))
        GL.Vertex3(Noued8.Coord(1), Noued8.Coord(2), Noued8.Coord(3))
        GL.Vertex3(Noued7.Coord(1), Noued7.Coord(2), Noued7.Coord(3))
        GL.End()
End Sub
```

# PDM Class

```
Imports System.Math
Public Class PDM
    Inherits Base
    Public E0, V, Fb0_Fc0, Kc, PsiDegre, Leq, Fck, Excent, SigmaT0, SigmaC, SigmaT, Dc, Dt, Damaged As
Double
    Public ac, bc, at, bt As Double

Public Function ConcreteDamagedPlasticityYieldCondition(stress() As Double, SigmaI As Double, SigmaII
As Double, SigmaIII As Double, Fb0_Fc0 As Double,SigmaC As Double, SigmaT As Double, Kc As Double) As
Double

        Dim J2, I1 As Double
        InvariantsContraintes(stress, I1,,, J2)
        Dim Alfa, Beta, Gamma As Double
        ParamatersCDP(Fb0_Fc0, SigmaC, SigmaT, Kc, Alfa, Beta, Gamma)
        Dim SigmaMax As Double
        SigmaMax = Max(Max(SigmaI, SigmaII), SigmaIII)
        If SigmaMax >= 0 Then
            Dim result As Double = (1 / (1 - Alfa))
            result = result * (Sqrt(3 * J2) + Alfa * I1 + Beta * SigmaMax) - SigmaC
            Return result
        Else
            Dim result As Double = (1 / (1 - Alfa))
            result = result * (Sqrt(3 * J2) + Alfa * I1 - Gamma * SigmaMax)
            result = result - SigmaC
            Return result
        End If
    End Function

Public Function DLambda(DStrain() As Double, DerivativeQ() As Double, DerivativeF() As
Double,Delastic(,) As Double, InElasticStrain As Double, SigmaTbarre As Double,SigmaMax As Double,
DInElasticStrain As Double) As Double

        Dim DDQ() As Double = MatriceVecteurMultipe(Delastic, DerivativeQ, 1)
        Dim DDE() As Double = MatriceVecteurMultipe(Delastic, DStrain, 1)
        Dim fc0 As Double = 0.4 * (Fck + 8)
        Dim DerivativeFStrain As Double = DerivativeYieldFunctionStrain(InElasticStrain, fc0,
SigmaTbarre, SigmaMax)
        Dim FirstScale As Double = 0
        Dim SecondScale As Double = 0
        Dim IH As Integer = DerivativeQ.Count - 1
        For i = 1 To IH
            FirstScale += DerivativeF(i) * DDE(i)
        Next
        FirstScale += DerivativeFStrain * DInElasticStrain
        For i = 1 To IH
            SecondScale += DerivativeF(i) * DDQ(i)
        Next
        Dim DLam As Double = FirstScale / SecondScale
        If DLam < 0 Then DLam = 0
        Return DLam
End Function

Public Function PlasticStressImplicit(Strain() As Double, Stress() As Double, Delastic(,) As
Double,StrainIncrement() As Double, InElasticStrain As Double, DInElasticStrain As Double, SigmaTbarre
As Double,SigmaMax As Double, alfa As Double) As Double()
        Dim a, b, Gamma, I1, J2, Theta, J3, SigmaI, SigmaII, SigmaIII As Double
        Dim Comprission As List(Of DoublePoint)
        Dim Tension As List(Of DoublePoint)
        SigmaT0 = 0.3016 * Fck ^ 0.6666667
        ValeurPropre(Stress, SigmaI, SigmaII, SigmaIII, Theta)
        InvariantsContraintes(Stress, I1,,, J2, J3)
        Dim DefPlasT, DefPlasC As Double
        DamageParametres(1, Strain, SigmaI, SigmaII, SigmaIII, Fck, Leq, bc, bt, SigmaC, SigmaT,
SigmaT0, Dc, Dt, Damaged, Comprission, Tension, E0, DefPlasT, DefPlasC)
        SigmaC = SigmaC / (1 - Dc)
        SigmaT = SigmaT / (1 - Dt)
        ParamatersCDP(Fb0_Fc0, SigmaC, SigmaT, Kc, a, b, Gamma)
        If SigmaI < 0 Then b = -Gamma
        Dim DerivativeF() As Double = DerivativeYieldFunctionStress(Stress, a, b, Theta, I1, J2, J3)
        Dim DerivativeQ() As Double = DerivativePotentialFunction(Stress, PsiDegre, Excent, SigmaT0)
```

```
        Dim Lamda As Double = DLambda(StrainIncrement, DerivativeQ, DerivativeF, Delastic,
InElasticStrain, SigmaTbarre, SigmaMax, DInElasticStrain)
        Dim IH As Integer = DerivativeQ.Count - 1
        For i = 1 To IH
            DerivativeQ(i) = DerivativeQ(i) * Lamda
        Next
        Return MatriceVecteurMultipe(Delastic, DerivativeQ)
End Function

Public Function YieldFunctionEstimation(Approch As Integer, Strain() As Double, Stress() As
Double,SigmaI As Double, SigmaII As Double, SigmaIII As Double, Fck As Double, Ieq As Double, bcBakhti
As Double, btBakhti As Double, Fb0_Fc0 As Double,Kc As Double, Compression As List(Of DoublePoint),
Tension As List(Of DoublePoint),ByRef SigmaC As Double, ByRef SigmaT As Double, ByRef SigmaT0 As
Double,ByRef Dc As Double, ByRef Dt As Double, ByRef D As Double,Optional Ebeton As Double = 0, ByRef
Optional DefPlasC As Double = 0) As Double

        Dim YieldFunction As Double = 0
        'Damaged Paramaters Evaluation
        Dim DefPlasT As Double
        Dim Resultats As Boolean = DamageParametres(Approch, Strain, SigmaI, SigmaII, SigmaIII, Fck,
Ieq, bcBakhti, btBakhti,
                                                    SigmaC, SigmaT, SigmaT0, Dc, Dt, D,
Compression, Tension, Ebeton, DefPlasT, DefPlasC)
        If Dc = 1 Then Dc = 0.999999999
        If Dt = 1 Then Dt = 0.999999999
        SigmaC = SigmaC / (1 - Dc)
        SigmaT = SigmaT / (1 - Dt)
        YieldFunction = ConcreteDamagedPlasticityYieldCondition(Stress, SigmaI, SigmaII, SigmaIII,
Fb0_Fc0, SigmaC, SigmaT, Kc)
        Return YieldFunction
End Function

Public Function DerivativeYieldFunctionStrain(InElasticStrain As Double, fc0 As Double, SigmaTBare As
Double, SigmaMax As Double) As Double
        Dim Resultats As Double = (Crocher(SigmaMax) / SigmaTBare) - 1
        Resultats = Resultats * fc0 * ac * (ac + 1) * (ac + 2) * bc * Exp(bc * InElasticStrain)
        Dim S As Double = ((2 * ac + 2) * Exp(bc * InElasticStrain) - ac) ^ 2
        Return Resultats / S
End Function

Public Function DerivativeYieldFunctionStress(Stress() As Double, a As Double, b As Double, Theta As
Double, I1 As Double, J2 As Double, J3 As Double) As Double()
        Dim Resultats(6) As Double
        Dim M1(0, 0), M2(0, 0), M3(6, 6), Flow(6, 6), x As Double
        FormM3D(Stress, M1, M2, M3)
        Dim J As Double = Sqrt(J2)
        Dim p As Double = I1 / 3
        Dim FirstSide() As Double = DerativeMeanStress()
        Dim SecondSide() As Double = DerativeDiviatoricStress(Stress, p, J)
        Dim ThirdSide() As Double = DerativeTheta(Stress, p, J, Theta, J3, M3)
        Dim DQ1, DQ2, DQ3 As Double
        DQ1 = (3 * a) / (1 - a) ' (3 * a + b) / (1 - a)
        Dim Sqrt3 As Double = Sqrt(3)
        DQ2 = 2 * b * Sin(Theta - 2 * PI / 3)
        DQ2 = Sqrt3 + DQ2 / Sqrt3
        DQ2 = Sqrt3 / (1 - a) 'DQ2 / (1 - a)
        DQ3 = 2 * b * J * Cos(Theta - 2 * PI / 3)
        DQ3 = 0 ' DQ3 / (Sqrt3 * (1 - a))
        For i = 1 To 6
            Resultats(i) = DQ1 * FirstSide(i) + DQ2 * SecondSide(i) + DQ3 * ThirdSide(i)
        Next
        Return Resultats
End Function

Public Function DerivativePotentialFunction(Stress() As Double, PsiDegre As Double, Epsilon As Double,
Sigmat0 As Double,Optional Type As Integer = 0) As Double()
   Select Case Type
            Case 0
                Dim TanPsi As Double = Tan(DegreToRadian(PsiDegre))
                Dim Resultats(6) As Double
                Dim M1(0, 0), M2(0, 0), M3(6, 6), Flow(6, 6), x As Double
                Dim DQ1, DQ2 As Double
                Dim J2 As Double
                InvariantsContraintes(Stress,,,, J2)
                DQ1 = TanPsi
```

```
                DQ2 = 3 / (2 * Sqrt((Epsilon * Sigmat0 * TanPsi) ^ 2 + 3 * J2))
                FormM3D(Stress, M1, M2, M3)
                For i = 1 To 6
                    For j = 1 To 6
                        Flow(i, j) = (M1(i, j) * DQ1 + M2(i, j) * DQ2)
                    Next
                Next
                For i = 1 To 6
                    x = 0
                    For j = 1 To 6
                        x += Flow(i, j) * Stress(j)
                    Next
                    Resultats(i) = x
                Next
                Return Resultats
            Case 1
                Dim TanPsi As Double = Tan(DegreToRadian(PsiDegre))
                Dim Resultats(6), I1, J2, J3 As Double
                InvariantsContraintes(Stress, I1,,, J2, J3)
                Dim J As Double = Sqrt(J2)
                Dim p As Double = I1 / 3
                Dim FirstSide() As Double = DerativeMeanStress()
                Dim SecondSide() As Double = DerativeDiviatoricStress(Stress, p, J)
                Dim DQ1, DQ2 As Double
                DQ1 = TanPsi
                DQ2 = 3 * J / Sqrt(3 * J ^ 2 + (Epsilon * Sigmat0 * TanPsi) ^ 2)
                For i = 1 To 6
                    Resultats(i) = DQ1 * FirstSide(i) + DQ2 * SecondSide(i)
                Next
                Return Resultats
        End Select
        Return Nothing
End Function

Public Function DamageParametres(Approch As Integer, Strain() As Double, SigmaI As Double, SigmaII As
Double,SigmaIII As Double, Fck As Double, Ieq As Double, bcBakhti As Double, btBakhti As Double, ByRef
SigmaC As Double,ByRef SigmaT As Double, ByRef SigmaT0 As Double, ByRef Dc As Double, ByRef Dt As
Double, ByRef D As Double, Comprission As List(Of DoublePoint), Tension As List(Of DoublePoint),
Ebeton As Double, ByRef DefPlasT As Double,ByRef DefPlasC As Double) As Boolean

        'According to B. Alfarah & al. 2017
        Select Case Approch
            Case 0  'Alfarah approach
                DamageParametres = True
                Dim r As Double = Rfunction(SigmaI, SigmaII, SigmaIII)
                Dim St, Sc As Double 'Must evaluate  Dc,Dt  'Eq 19 & 20
                Dim Hc, Ht As Double
                Dim StrI, StrII, StrIII As Double
                SigmaT = 0
                SigmaT0 = 0
                SigmaC = 0
                Dc = 0
                Dt = 0
                D = 0
                ValeurPropre(Strain, StrI, StrII, StrIII)
                Dim DefT As Double = r * Max(Max(StrI, StrII), StrIII)
                Dim DefC As Double = -(1 - r) * Min(Min(StrI, StrII), StrIII)
                Dim DefCpl, DefTpl As Double
             Hc = 0.9
                Ht = 0
                Dim b As Double = 0.9
                Dim Newb As Double
                Dim Fcm, Ftm, Fc0, Ft0, Defcm, Deftm, Eci, E0, Gf, Gch, Wc As Double
                Fcm = Fck + 8
                Ftm = 0.3016 * Fck ^ (2 / 3)
                Defcm = 0.0007 * Fcm ^ 0.31
                Eci = 10000 * Fcm ^ (1 / 3)
                E0 = Eci * (0.8 + 0.2 * (Fcm / 88))
                Gf = 0.073 * Fcm ^ 0.18
                Gch = Gf * (Fcm / Ftm) ^ 2
                Wc = 5.14 * Gf / Ftm
                Deftm = Ftm / E0
                Fc0 = 0.4 * Fcm
                Ft0 = Ftm
                Dim ac, at, bc, bt As Double
```

```
                ac = 7.873
                at = 1
                bc = (1.97 * Fcm / Gch) * Ieq
                bt = (0.453 * Fck ^ (2 / 3) / Gf) * Ieq
                Sc = 1 - Hc * (1 - r)
                St = 1 - Ht * r
                Dim ElaticDefor As Double = 0.4 * Fcm / E0
                Dim iter As Integer = 0
                Do 'Lancer iteration
                    iter += 1
                    If DefC >= 0 And DefC <= ElaticDefor Then
                        SigmaC = SigmaCI(DefC, E0)
                        DefPlasC = 0
                        '   If DefC <> 0 Then DamageParametres = False
                    ElseIf DefC > ElaticDefor And DefC <= Defcm Then
                        SigmaC = SigmaCII(DefC, Defcm, Fcm, Eci)
                        DefPlasC = DefC - SigmaC / E0
                    ElseIf DefC > Defcm Then
                        SigmaC = SigmaCIII(DefC, Defcm, Fcm, Gch, Ieq, b, E0)
                        DefPlasC = DefC - SigmaC / E0
                    End If
                    If DefT >= 0 And DefT <= Deftm Then
                        SigmaT = SigmaTI(DefT, E0)
                        DefPlasT = 0
                        '   If DefT <> 0 Then DamageParametres = False
                    ElseIf DefT > Deftm Then
                        SigmaT = SigmaTII(DefT, Deftm, Ieq, Wc, Ftm)
                        DefPlasT = DefT - SigmaT / E0
                    End If
                    Dc = 1 - (1 / (2 + ac)) * (2 * (1 + ac) * Exp(-bc * DefPlasC) - ac * Exp(-2 * bc *
DefPlasC))
                    Dt = 1 - (1 / (2 + at)) * (2 * (1 + at) * Exp(-bt * DefPlasT) - at * Exp(-2 * bt *
DefPlasT))
                    D = 1 - (1 - St * Dc) * (1 - Sc * Dt)
                    If Dc < 0 Then Dc = 0
                    If Dt < 0 Then Dt = 0
                    If D < 0 Then D = 0
                    If DefPlasC = 0 Then
                        SigmaC = Fc0
                        DefCpl = 0
                    Else
                        DefCpl = DefPlasC - SigmaC * Dc / (E0 * (1 - Dc))
                        If DefCpl < 0 Then DefCpl = 0
                    End If
                    If DefPlasT = 0 Then
                        SigmaT = Ft0
                        DefTpl = 0
                    Else
                        DefTpl = DefPlasT - SigmaT * Dt / (E0 * (1 - Dt))
                    End If
                    If DefPlasC <> 0 Then
                        Newb = DefCpl / DefPlasC
                        If Convergence(Newb, b, 0.0001) Or iter > 400 Then
                            Exit Do
                        End If
                        b = Newb
                    Else
                        Exit Do
                    End If
                Loop
                SigmaT0 = SigmaTI(Deftm, E0)
            Case 1 'According to Bakhti approach
                DamageParametres = True
                Dim r As Double = Rfunction(SigmaI, SigmaII, SigmaIII)
                If SigmaI > 0 Then
                    SigmaI += 0
                End If
                Dim St, Sc As Double 'Must evaluate  Dc,Dt  'Eq 19 & 20
                Dim Hc, Ht As Double
                Dim StrI, StrII, StrIII As Double
                SigmaT = 0
                SigmaT0 = 0
                SigmaC = 0
                Dc = 0
                Dt = 0
```

```
                D = 0
                ValeurPropre(Strain, StrI, StrII, StrIII)
                Dim DefT As Double = r * Max(Max(StrI, StrII), StrIII)
                Dim DefC As Double = -(1 - r) * Min(Min(StrI, StrII), StrIII)
                Dim SigT As Double = r * Max(Max(SigmaI, SigmaII), SigmaIII)
                Dim SigC As Double = -(1 - r) * Min(Min(SigmaI, SigmaII), SigmaIII)
                Hc = 0.9
                Ht = 0
                Dim Fcm, Fc0, Ft0, Ftm, Defcm, Deftm, Eci, E0, Gf, Gch, Wc As Double
                Fcm = Fck + 8
                Ftm = 0.3016 * Fck ^ (2 / 3)
                Defcm = 0.0007 * Fcm ^ 0.31
                Eci = 10000 * Fcm ^ (1 / 3)
                E0 = Eci * (0.8 + 0.2 * (Fcm / 88))
                Gf = 0.073 * Fcm ^ 0.18
                Gch = Gf * (Fcm / Ftm) ^ 2
                Wc = 5.14 * Gf / Ftm
                Deftm = Ftm / E0
                Fc0 = 0.4 * Fcm
                Ft0 = Ftm
                Dim ac, at, bc, bt As Double
                ac = 7.873
                at = 1
                bc = bcBakhti
                bt = btBakhti
                Sc = 1 - Hc * (1 - r)
                St = 1 - Ht * r
                DefPlasC = FindInelasticStrain(0, 0.05, 0.00001, Fc0, ac, bc, E0, DefC)
                DefPlasT = FindInelasticStrain(0, 0.005, 0.000001, Ft0, at, bt, E0, DefT)
                Dc = 1 - (2 * (1 + ac) * Exp(-bc * DefPlasC) - ac * Exp(-2 * bc * DefPlasC)) / (2 +
ac)
                Dt = 1 - (2 * (1 + at) * Exp(-bt * DefPlasT) - at * Exp(-2 * bt * DefPlasT)) / (2 +
at)
                D = 1 - (1 - St * Dc) * (1 - Sc * Dt)
                If DefPlasC = 0 Then
                    SigmaC = Fc0
                Else
                    SigmaC = Fc0 * ((1 + ac) * Exp(-bc * DefPlasC) - ac * Exp(-2 * bc * DefPlasC))
                End If
                If DefPlasT = 0 Then
                    SigmaT = Ft0
                Else
                    SigmaT = Ft0 * ((1 + at) * Exp(-bt * DefPlasT) - at * Exp(-2 * bt * DefPlasT))
                End If
                SigmaT0 = SigmaTI(Deftm, E0)
                If Dc < 0 Then Dc = 0
                If Dt < 0 Then Dt = 0
                If D < 0 Then D = 0
            Case 2 'From curves
                DamageParametres = True
                Dim r As Double = Rfunction(SigmaI, SigmaII, SigmaIII)
                Dim St, Sc As Double 'Must evaluate  Dc,Dt  'Eq 19 & 20
                Dim Hc, Ht As Double
                Dim StrI, StrII, StrIII As Double
                SigmaT = 0
                SigmaT0 = 0
                SigmaC = 0
                Dc = 0
                Dt = 0
                D = 0
                ValeurPropre(Strain, StrI, StrII, StrIII)
                Dim DefT As Double = r * Max(Max(StrI, StrII), StrIII)
                Dim DefC As Double = -(1 - r) * Min(Min(StrI, StrII), StrIII)
                Hc = 0.9
                Ht = 0
                Sc = 1 - Hc * (1 - r)
                St = 1 - Ht * r
                DefPlasC = FindInelasticStrainFromCurve(DefC, Comprission, Ebeton)
                DefPlasT = FindInelasticStrainFromCurve(DefT, Tension, Ebeton)
                SigmaC = FindStressFromCurve(DefPlasC, Comprission, Dc)
                SigmaT = FindStressFromCurve(DefPlasT, Tension, Dt, SigmaT0)
                D = 1 - (1 - St * Dc) * (1 - Sc * Dt)
                If Dc < 0 Then Dc = 0
                If Dt < 0 Then Dt = 0
                If D < 0 Then D = 0
```

```
        End Select
End Function

Public Function FindInelasticStrain(IntrevalStart As Double, IntrevalEnd As Double, Precesion As
Double,Fc0 As Double, ac As Double, bc As Double, E0 As Double, TotalStrain As Double) As Double
        Dim IntervalMidde As Double = (IntrevalStart + IntrevalEnd) / 2
        'Find total strain intervale
        Dim TotalIntrevalStart, TotalIntrevalEnd, TotalMiddelInterval As Double
        Dim SigmaC1, SigmaC2, SigmaC3 As Double
        SigmaC1 = Fc0 * ((1 + ac) * Exp(-bc * IntrevalStart) - ac * Exp(-2 * bc * IntrevalStart))
        SigmaC2 = Fc0 * ((1 + ac) * Exp(-bc * IntervalMidde) - ac * Exp(-2 * bc * IntervalMidde))
        SigmaC3 = Fc0 * ((1 + ac) * Exp(-bc * IntrevalEnd) - ac * Exp(-2 * bc * IntrevalEnd))
        TotalIntrevalStart = IntrevalStart + SigmaC1 / E0
        TotalMiddelInterval = IntervalMidde + SigmaC2 / E0
        TotalIntrevalEnd = IntrevalEnd + SigmaC3 / E0
        If TotalStrain < TotalIntrevalStart Then
            Return 0
        End If
        If Abs(TotalStrain - TotalIntrevalStart) < Precesion Then
            Return IntrevalStart
        End If
        If Abs(TotalStrain - TotalMiddelInterval) < Precesion Then
            Return IntervalMidde
        End If
        If Abs(TotalStrain - TotalIntrevalEnd) < Precesion Then
            Return IntrevalEnd
        End If
        If TotalStrain > TotalIntrevalStart And TotalStrain < TotalMiddelInterval Then
            Return FindInelasticStrain(IntrevalStart, IntervalMidde, Precesion, Fc0, ac, bc, E0,
TotalStrain)
        End If
        If TotalStrain > TotalMiddelInterval And TotalStrain < TotalIntrevalEnd Then
            Return FindInelasticStrain(IntervalMidde, IntrevalEnd, Precesion, Fc0, ac, bc, E0,
TotalStrain)
        End If
        Return Nothing
    End Function
End Class
```

نسخة نهائية مصححة وقابلة للايداع