# Implementation of an elasto-plastic model for non-cohesive soils in the FEniCS finite element library

Samadrita Karmakar

August 2019

# Contents

# Chapter 1

# Introduction

Poro-Mechanics is the study of materials which are influenced by the existence of pore fluid within the material. This influences the mechanical behaviour of material.
One of the most common material where Poro-Mechanics is used to predict behaviour, is that of soils. A continuum approach is taken to take into consideration the discontinuities that exists within the soil. (To be changed)

# Chapter 2

# Plasticity

When a material deforms reversibly, its mechanical behaviour is defined as elastic. In general, there exists a particular domain of mechanical states within which the material behaves elastically. The material will deform reversibly within this domain, defined as the elastic domain and is bounded by a limit called elastic limit or yield surface [5]. Within the elastic domain, if the material is unloaded, the material will return to its original configuration. Considering only principal stresses, in 1-D case, the elastic limit is a point is known as a yield point ($\sigma_y$). In 2-D case the elastic limit is in form of a curve and in 3-D case the elastic limit is in form of a surface. This surface is defined as yield surface [4] or loading surface [5]. All ideas introduced in this chapter are from [4] or from [5] and form the basis on which this work has been executed.

From the yield surface the material no more remains elastic and a part of the deformation becomes non-reversible. The elastic limit mentioned above starts evolving with further strains and plasticity theories try to model this behaviour.

Any strain from the elastic limit is generally consist a reversible and a non-reversible strains. Any further strain from the elastic limit are non-reversible and is called plastic strain. If the hypothesis of small strains holds true, then the total strain ($\boldsymbol{\epsilon}$) in the body can be decomposed into two kind of strains. Elastic strain ($\boldsymbol{\epsilon}^e$) and plastic strains ($\boldsymbol{\epsilon}^p$). This can be expressed by the following equation:

$$\boldsymbol{\epsilon} = \boldsymbol{\epsilon^e} + \boldsymbol{\epsilon^p} \tag{2.1}$$

The stress-strain linear elastic relationship in its broadest form is known as the generalized Hooke's law which is given by the following equations [1]:

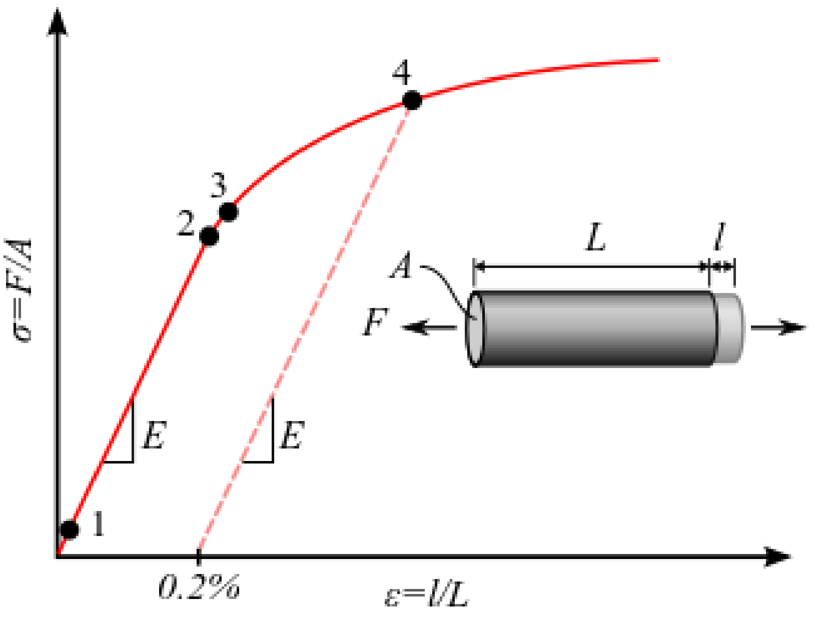

Figure 2.1: Stress–strain curve showing typical yield behavior for nonferrous alloys [7]

$$\boldsymbol{\sigma} = \mathbb{C} : \boldsymbol{\epsilon^e} \tag{2.2}$$

By using the equation 2.1, we can write the equation 2.2 as,

$$\boldsymbol{\sigma} = \mathbb{C} : (\boldsymbol{\epsilon} - \boldsymbol{\epsilon^p}) \tag{2.3}$$

As shown in figure 2.1, as the load is applied, the stress reaches the yield point at 3 and evolves to point 4. , The stress corresponding to the point 4 becoming the new elastic limit. If the load is unloaded now, the stress reduces to zero and the reversible part of the strain goes back to zero but the plastic part remains. If again loaded back, the stress now acts elastic up to the new elastic limit. This new elastic limit due to the evolution of the yield surface which over here in 1-D is a point.

## Yield Function

To know whether the material is in the elastic domain or at the yield surface, a function called the yield function, yield criterion or loading function,(usually abbreviated as $f$) is used. Where $\boldsymbol{\sigma}$ is the Cauchy Stress Tensor (Simply, stress tensor) and $\boldsymbol{q}$ is also commonly known as Hardening Force. Yield function returns scalar value which informs of of the configuration of the mechanical state $(\boldsymbol{\sigma}, \boldsymbol{q})$ with respect to the loading surface in the material. Mathematically this is expressed as, $f : \mathbb{S} \times \mathbb{R}^m \rightarrow \mathbb{R}$ [5]

## Elastic Domain

We now wish to define the elastic domain,the space of plastically admissible mechanical states $(\boldsymbol{\sigma}, \boldsymbol{q})$, which is supposed to evolve when mechanical state $(\boldsymbol{\sigma}, \boldsymbol{q})$ reaches it's boundary. [5]

$$E_{\boldsymbol{\sigma}} := \{(\boldsymbol{\sigma}, \boldsymbol{q}) \in S \times R^m | f \leq 0\} \tag{2.4}$$

The elastic domain corresponding to the interior of $E_\sigma$ is,

$$int(E_{\boldsymbol{\sigma}}) := \{(\boldsymbol{\sigma}, \boldsymbol{q}) \in S \times R^m | f < 0\} \tag{2.5}$$

and the yield surface, or loading surface, is defined as

$$\partial E_{\boldsymbol{\sigma}} = \{(\boldsymbol{\sigma}, \boldsymbol{q}) \in S \times R^m | f = 0\} \tag{2.6}$$

Any set of values of $(\boldsymbol{\sigma}, \boldsymbol{q})$ that does not fall within the Elastic domain of $E_{\boldsymbol{\sigma}}$ are "non admissible and are ruled out in classical plasticity" [5]

## Plstic Strain and Hardening Force

As from the equation 2.1 we see that total strain is the sum of elastic strain and plastic strain. Plastic strain is generally evolutionary, i.e., it follows generally a flow rule, expressed as following:

$$\dot{\boldsymbol{\epsilon}}^p = \dot{\lambda} \boldsymbol{r}(\boldsymbol{\sigma}, \boldsymbol{q}) \tag{2.7}$$

where $\dot{\lambda}$ is a called plastic multiplier and $\boldsymbol{r}$ is the flow potential.The hardening force also evolves with time and can be expressed as [5],

$$\dot{\boldsymbol{q}} = -\dot{\lambda} \boldsymbol{H}(\boldsymbol{\sigma}, \boldsymbol{q}) \tag{2.8}$$

## Kuhn Tucker Condition

Consider that $E_{\boldsymbol{\sigma}}$ is supposed to evolve only when the stress reaches the yield surface, hence plastic strain and hardening force evolves. There should be no evolution of plastic strain and hardening force when the stress is not at the yield surface. Considering this, the *Kuhn Tucker* complementary condition seem to be a natural conclusion, i.e., $\dot{\lambda} > 0$ only when $f(\boldsymbol{\sigma}, \boldsymbol{q}) = 0$. Mathematically [5],

$$\dot{\lambda} \geq 0,\ f(\boldsymbol{\sigma}, \boldsymbol{q}) \leq 0 \tag{2.9}$$

and

$$\dot{\lambda} f(\boldsymbol{\sigma}, \boldsymbol{q}) = 0 \tag{2.10}$$

There also exists the consistency condition, $\dot{\lambda}\dot{f}(\boldsymbol{\sigma}, \boldsymbol{q}) = 0$ (2.11)

The interpretation can be simplified as [5],

$$if,\ f < 0 \longleftrightarrow (\boldsymbol{\sigma}, \boldsymbol{q}) \in\ int(E_\sigma = 0) \text{ (elastic)} \tag{2.12}$$

$$if,\ f = 0,\ \ \longleftrightarrow (\boldsymbol{\sigma}, \boldsymbol{q}) \in\ \partial E_\sigma = 0, \&\ \dot{f} < 0,\ \dot{\lambda} = 0 \text{ (elastic unloading)} \tag{2.13}$$

$$if,\ f = 0,\ \ \longleftrightarrow (\boldsymbol{\sigma}, \boldsymbol{q}) \in\ \partial E_\sigma = 0, \&\ \dot{f} = 0,\ \dot{\lambda} = 0 \text{ (neutral loading)} \tag{2.14}$$

$$if,\ f = 0,\ \ \longleftrightarrow (\boldsymbol{\sigma}, \boldsymbol{q}) \in\ \partial E_\sigma = 0, \&\ \dot{f} = 0,\ \dot{\lambda} \geq 0 \text{ (plastic loading)} \tag{2.15}$$

The definition of $\boldsymbol{r}(\boldsymbol{\sigma}, \boldsymbol{q})$ as defined in equation 2.7 can be defined in the following ways,

$$\dot{\boldsymbol{\epsilon}}^p = \dot{\lambda}\frac{\partial f(\boldsymbol{\sigma}, \boldsymbol{q})}{\partial \boldsymbol{\sigma}} \tag{2.16}$$

In the above case, the plastic strain is directly dependent on the yield function, it is known as *associative plasticity.*

It might also be the case, that the plastic strain is not dependent directly on the yield function, then it is known as *non-associative plasticity* and generally expressed as,

$$\dot{\boldsymbol{\epsilon}}^p = \dot{\lambda}\frac{\partial g(\boldsymbol{\sigma}, \boldsymbol{q})}{\partial \boldsymbol{\sigma}} \tag{2.17}$$

where $g(\boldsymbol{\sigma}, \boldsymbol{q})$ is a function that returns a scalar and may be defined mathematically as, $g : \mathbb{S} \times \mathbb{R}^m \to \mathbb{R}$. The function is known as *plastic potential.* [4]

## A Simple Introduction to Return Mapping

The evolution of yield surface is determined numerically by using an algorithm known as “Return Mapping algorithm” [5]. It ensures the current mechanical state to remain in the space of admissible state ($E_{\boldsymbol{\sigma}}$).
Here an algorithm for associative perfect plasticity (i.e. associative plasticity without hardening processes) is discussed following [5] and is the principal inspiration for this work.

Since, plastic strain evolves with loading, the current plastic strain may be evaluated using backward euler method and can be stated as, Step 1: Compute the trial stress but plugging in the total strain, to $\sigma_{n+1}^{trial} = \mathbb{C} : (\boldsymbol{\epsilon}_{n+1} - \boldsymbol{\epsilon}_n^p)$

Step 2: Compute the $f_{n+1}^{trial}(\boldsymbol{\sigma})$ by plugging in the value of $\boldsymbol{\sigma}_{trial}$

Step 3: If $f_{n+1}^{trial}(\boldsymbol{\sigma}) > 0$ then go the next Step, or else go to Step 1 to compute the new total strain.

Step 4: Now that $f_{n+1}^{trial}(\boldsymbol{\sigma}) > 0$, return mapping has to be performed to set $f_{n+1}(\boldsymbol{\sigma}) = 0$ and $\Delta\lambda$ is $> 0$ A new plastic strain is calculated by preforming backward euler on the equation 2.16

$$\boldsymbol{\epsilon}_{n+1}^p = \boldsymbol{\epsilon}_n^p + \Delta\lambda \frac{\partial f_{n+1}}{\partial \boldsymbol{\sigma}} \tag{2.18}$$

Since Newton Iteration has to be performed, we may write,

$$R_{n+1} = -\boldsymbol{\epsilon}_{n+1}^p + \boldsymbol{\epsilon}_n^p + \Delta\lambda \frac{\partial f_{n+1}}{\partial \boldsymbol{\sigma}} \tag{2.19}$$

We perform Taylor expansion on 2.19 to find the value of $\Delta\lambda$ such that.

$$R_{n+1}^{(k+1)} = R_{n+1}^{(k)} + \frac{\partial R_{n+1}^{(k)}}{\partial \Delta\lambda} d(\Delta\lambda) \tag{2.20}$$

from the equation 2.3 we can write,

$$\mathbb{C}^{-1} : \boldsymbol{\sigma}_{n+1} = \boldsymbol{\epsilon}_{n+1} - \boldsymbol{\epsilon}_{n+1}^p \tag{2.21}$$

Using the above equation and equation 2.20 we can write,

$$R_{n+1}^{(k+1)} = R_{n+1}^{(k)} + \left[\mathbb{C}^{-1} + \frac{\partial^2 f_{n+1}}{\partial \boldsymbol{\sigma}^2}\Delta\lambda\right] \frac{\partial \sigma_{n+1}}{\partial \Delta\lambda} d(\Delta\lambda) + \frac{\partial f_{n+1}}{\partial \boldsymbol{\sigma}} d(\Delta\lambda) \tag{2.22}$$

Since Yield function is only dependent on $\sigma_{n+1}$, we can write, $f_{n+1} := f(\boldsymbol{\sigma}_{n+1})$

now, considering Taylor Expansion of Yield function, we have,

$$f_{n+1}^{k+1} = f_{n+1}^k + \frac{\partial f_{n+1}^k}{\partial \boldsymbol{\sigma}_{n+1}} \frac{\partial \boldsymbol{\sigma}_{n+1}}{\partial \Delta\lambda^k} d(\Delta\lambda^k) \tag{2.23}$$

where $f_{n+1}^{k+1} = 0$ on convergence and so does $R_{n+1}^{(k+1)} = 0$ in equation 2.22. Substituting the value of $\frac{\partial \sigma_{n+1}}{\partial \Delta\lambda} d(\Delta\lambda)$ from equation 2.22 into equation 2.23, we get,

$$\left[\mathbb{C}^{-1} + \frac{\partial^2 f_{n+1}}{\partial \boldsymbol{\sigma}_{n+1}^2}\Delta\lambda\right]^{-1} \left(\left[R_{n+1}^{(k)}\right] + \left[\frac{\partial f_{n+1}}{\partial \boldsymbol{\sigma}}\right] d(\Delta\lambda^k)\right) = f_{n+1}^k \tag{2.24}$$

Hence,

$$\rightarrow d(\Delta\lambda^k) = \frac{f^k_{n+1} - \frac{\partial f_{n+1}}{\partial \sigma_{n+1}} : [\boldsymbol{\Xi}] : R^{(k)}_{n+1}}{\left[\frac{\partial f_{n+1}}{\partial \boldsymbol{\sigma}}\right] : [\boldsymbol{\Xi}] : \frac{\partial f_{n+1}}{\partial \boldsymbol{\sigma}}} \tag{2.25}$$

where $\boldsymbol{\Xi} = \left[\mathbb{C}^{-1} + \frac{\partial^2 f_{n+1}}{\partial \boldsymbol{\sigma}^2_{n+1}}\Delta\lambda\right]^{-1}$

The value of $d(\Delta\lambda^k)$ is stored for the next step.

Step 5: Calculate the new value of $\frac{\partial \boldsymbol{\sigma}_{n+1}}{\partial \Delta\lambda} d(\Delta\lambda)$

$$\frac{\partial \boldsymbol{\sigma}_{n+1}}{\partial \Delta\lambda^k} d(\Delta\lambda) = \boldsymbol{\Xi} : \left[-R^{(k)}_{n+1} - \frac{\partial f^k_{n+1}}{\partial \boldsymbol{\sigma}_{n+1}}\right] \tag{2.26}$$

Step 6: Use the value of $\frac{\partial \boldsymbol{\sigma}_{n+1}}{\partial \Delta\lambda^k} d(\Delta\lambda)$ to find $\Delta\epsilon^p_{n+1}$

$$\Delta\epsilon^{p(k)}_{n+1} = -\mathcal{C} : \frac{\partial \boldsymbol{\sigma}_{n+1}}{\partial \Delta\lambda} d(\Delta\lambda) \tag{2.27}$$

Step 7: Next find the new values of $\epsilon^{p(k)}_{n+1}$ and $\Delta\lambda^k$

$$\epsilon^{p(k+1)}_{n+1} = \epsilon^{p(k)}_{n+1} + \Delta\epsilon^{p(k)}_{n+1} \tag{2.28}$$
$$\Delta\lambda^{k+1} = \Delta\lambda^k + d(\Delta\lambda^k) \tag{2.29}$$

A more complex and implementation has been used in this master thesis for the return mapping algorithm. All the algorithm from the main weak forms used to the actual return mapping algorithm used, shall be discussed in the chapter "Implementation."

# Chapter 3

# FEniCS

The Finite Element library used in the execution of this Master Thesis is from the FEniCS Project [6]. FEniCS Project automates the solution of mathematical models based on differential equations [2].

The FEniCS Project started in 2003 with an idea to automate the solution of mathematical models based on differential equations. FEniCS has interfaces for both in C++ and in Python. [2]

The FEniCS libraries used in the execution of this master Thesis is the DOLFIN interface and the FFC compiler. For problems related to that of Solid Mechanics, an extensions of the FEniCS project had to be used. This is mostly due to the requirement of being able to access the yield function, at each and every gauss point. Also, at each gauss point the plastic tangent tensor has to be calculated. This operation may be achieved by using the FEniCS Solid Mechanics library which is an extension to the FEniCS project.

The FEniCS Solid Mechanics library mentioned above was developed by Kristian B. Ølgaard and Garth N. Wells. The mainline of the library was actively developed between 2013 to 2017 but development since has been slow. Much of the newer developments have been on other branches but not merged to the mainline branch. It may be so because they are not ”production ready” or that the original developers of the library are not available for maintenance.

On investigating the capabilities of the FEniCS Solid Mechanics library, it was found that the library was designed to work for simpler plasticity like $J2$ models. Hence the most complex part of the Thesis was to extend the library capabilities well enough, such that it is possible to execute more complex plasticity models such as Sinfonietta Classica without breaking backward compatibility to already implemented simpler models such as $J2$. More details on how and what are the changes made to the library has been mentioned in detail in the ”Implementation” Chapter. In the rest of this chapter a very brief introduction to the FEniCS project has been provided.

## DOLFIN

DOLFIN is a C++/Python library that functions as the main user interface of FEniCS. DOLFIN has dual function in the FEniCS project. It is a core component of the FEniCS project. In addition, DOLFIN also acts as the user interface of FEniCS. All communication between a user program, other core components of FEniCS and external software is routed through wrapper layers that are implemented as part of the DOLFIN user interface. When using the C++ interface, the variational forms are expressed in UFL form language. [2]

In Finite element codes, many parts of the Finite Element code can be reused. These may be parts such as the assembly of local matrices to global matrices, mesh information or matrix manipulation. The issue that arises is generally with the formation of the local contribution or the local stiffness matrix per element. This local stiffness matrix is a result of the weak form of the given problem. The weak form is depended on the mathematical model that is being solved. This part generally has to be programmed by the application developer and is unique to every different mathematical model. The FEniCS Project attempts to generalize this part of the algorithm by introducing a Python based language, the ”Unified Form Language” or in abbreviated form, the UFL. [3]

For the C++ interface, this UFL file generated by the application developer is converted to a low level UFC code using the ”Fenics Form Compiler” (FFC) that can be imported into a C++ file. Then using the DOLFIN interface, the global stiffness matrices may be assembled.

# Bibliography

[1] Eduardo W.V. Chaves, *Notes on Continuum Mechanics, 2013*

[2] *Automated Solution of Differential Equations by the Finite Element Method - The FEniCS Book, 2011* Edited by Anders Logg, Kent-Andre Mardal, Garth N. Wells

[3] *UFC Specification and User Manual 1.1 February 21, 2011*

[4] Ronaldo I. Borja *Plasticity Modeling & Computation*

[5] J.C. Simo, T.J.R. Huges *Computational Inelasticity*

[6] *https://fenicsproject.org/*

[7] *Plasticity (Wikipedia)*