An investigation into aspects of rate-independent single crystal plasticity

Norman J. Richardson

Thesis Presented in partial fulfillment
for the Degree of
MASTER OF SCIENCE IN ENGINEERING
in the Department of Mechanical Engineering
UNIVERSITY OF CAPE TOWN

February 12, 2012

Plagiarism declaration

I know the meaning of plagiarism and declare that all of the work in the document, save for that which is properly acknowledged, is my own.

Signed: _____

Date: February 12, 2012

Abstract

The objective of the dissertation is to compare various algorithms for rate-independent single-crystal plasticity restricted to the infinitesimal case. The rate-independent case presents significant numerical challenges such as ill-conditioning and non-smoothness. The majority of the algorithms in the literature are heuristic and their accuracy, stability and performance not immediately clear. Other algorithms are more mathematically sound but computationally inefficient. This dissertations aims to provide a clear understanding of the algorithms, their implementation within a non-linear finite element code, and a comparison of the computational efficiency.

In order to compare the various algorithms, certain background material is presented; this includes discussions on the derivation of the constitutive relations, mathematical treatments, numerical techniques, and programing practices.

Once the background material for the various rate-independent algorithms has been introduced, the focus turns to a series of numerical "experiments". The initial objective is to establish if the various algorithms produce results that agree with one another and with those available in the literature. The second objective is to qualitatively measure the performance of the various algorithms for a challenging problem. The outcome of the numerical experiments indicate that all algorithms produce essentially the same results but there is a large variation in efficiency.

The algorithms are compared to those in the literature using an experiment presented in [29]. This involves the shearing of a perfectly plastic cube. This experiment is repeated for several orientations of the crystal structure. Results for the slip versus the strain are indistinguishable from those in the literature.

A second experiment, also presented in [29], illustrates that the algorithms produce a similar macroscopic phenomena known as slip bands. This experiment involves placing a rectangular strip of isotropically hardening material in tension. To induce the development of slip bands, an artificial weakness is introduced into the material. The results for the various algorithms compare well with one another.

The second experiment quantifies the performance of the various algorithms. It is found that the equivalent algorithm, as presented in [37], is the least efficient. The set search algorithm,

presented in [29], is the most efficient. These two algorithms share the same mathematical formulation, yet the equivalent algorithm is approximately 18-20 times less efficient. The augmented Lagrangian algorithm, as presented in [37], is the third most efficient algorithm. This algorithm is more than twice as efficient as the equivalent algorithm, but approximately seven times less efficient than the set search algorithm. The modified augmented Lagrangian, a modified version of the algorithm presented [37], performed slightly better than augmented Lagrangian algorithm. The modification lead to an increase in efficiency of approximately 17%, but it is approximately six times less efficient than the set search algorithm.

Finally, a further analysis of the performance is conducted. The goal is to confirm that the performance results are due to a generally better performing solution scheme, and not due to anomalous events. It is found, in all cases, that the improvement in efficiency is indeed due to a general increase in performance.

Acknowledgement

I gratefully acknowledge the guidance, encouragement, and patience of my supervisor Prof. Daya Reddy and cosupervisor Dr. Andrew McBride. I am greatly thankful that they continued to advice me, even though for a large majority of the time this was a long distance relationship. I am also very grateful for the work environment that Prof has created at CERECAM and for the work colleagues I have found there. Many of these colleagues have been instrumental in the completion of this work, in particular Beverley Grieshaber, Jean-Paul Pelteret and Helen Morrissey. I thank you for your willingness to listern to me vocalise my concerns and even help me solve many problems. I am also very grateful for the support of my friends in Cape Town, in particular Ryan Harmuth, Anton Eicher, and Shakira Maharaj. A particular mention should be made for Shakira for being my dedicated proof reader and attentive listener. Thanks to her effort this thing became legible and comprehendable. I will never subject you to ramblings about crystal plasticity again. I would like to thank my family. I thank my parents, Beverely Stone and Marshall Richardson, for their emotional and financial support, and for their general guidance and encouragment. I also thank my brothers - Victor, James, Marshall and Louis Richardson - who I live life alongside. Finally, I dedicate this thesis to my mother.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The classical theory of elastoplasticity has been successful at describing a wide range of behaviour for inelastic materials. The theory is well established and is widely used to model the macroscopic behaviour of polycrystalline materials. However, the classical theory is limited in describing behaviour in which microstructural effects are important.

It is well understood that the specific behaviour of a material is strongly related to its underlying microstructure. In some cases the microstructural response can have a profound effect on the macroscopic mechanical behaviour of the material. The material class we shall focus on is that of metal single crystals. There are many reasons why the behaviour of these materials is of interest. For instance, this theory is essential to understanding the development of a phenomenon in polycrystals called texturing [33]. These materials have also found their way into industrial applications which require large single crystals, most notably for use as turbine blades in aircraft engines [4]. How these materials respond to external loading is of practical importance and interest.

There are two major classifications for flow behaviour, viz. viscoplastic or rate-dependent, and rate-independent. There exist many computational treatments of viscoplastic crystal plasticity, see for example [33, 31, 6, 32, 40, 36, 27]. The rate-dependent treatment of rate-independent problems is attractive because it negates many of the inherent computational issues of rate-independent problems. However, it has been suggested by Miehe [28] that rate-dependent methods have at times been applied in conditions under which rate-independence is the more appropriate assumption. This treatment is justified by assuming that the

material is only marginally viscoplastic.

At the heart of rate-independent single-crystal plasticity are the inherent problems of non-uniqueness, ill-conditioning, and non-smoothness. The problem of non-uniqueness results in a non-unique choice of active sets, as illustrated by Taylor [41], and a potentially non-invertible set of flow rules. The cause of this non-uniqueness can be traced back to the crystal structure itself. Most notably, crystals having a face-centered cubic (f.c.c.) and body centered cubic (b.c.c.) structure exhibit this feature.

Once formulated as an incremental problem, after time-discretisation has been introduced, the flow relation can be cast as a constrained optimisation problem. This is immediately evident from the principle of maximum plastic dissipation. Several classical treatments of constrained optimisation problems can be found in Luenberger [26] including Lagrange multipliers, penalty methods, and the augmented Lagrangian formulation. The application of each treatment results in a different set of flow rules which leads to adopting different algorithms to solve the problem. It should be noted that the treatments are equivalent except for the penalty method which is an approximation.

Miehe [28] has approached the problem using Lagrange multipliers which leads to an active set search. The active set search has its own set of problems. One problem is that the criteria used in the search are not explicitly given by the constitutive relations. Over time Miehe has modified and adapted the criteria, for instance compare [29, 30, 28]. The second issue is that the system of equations remains potentially non-invertible. Miehe has addressed this issue using standard methods, these include the methods of singular value decomposition, pseudo-inverse, and Moore–Penrose pseudo-inverse [29, 28].

Schmidt-Baldassari [37] has approached the problem using an augmented Lagrangian formulation which leads to a fixed point algorithm for the flow rules. This approach can be seen as solving a series of rate-dependent problems that converge to the rate-independent formulation in the limit. It negates the issues of non-uniqueness of the active set as well as the non-invertibility of the flow rule.

The Lagrange multiplier treatment does not necessarily lead to a set search approach. Instead, the complementarity conditions can be recast in an equation that enforces the conditions, see [37]. This set of equations is then solved by the use of a Newton–Raphson approach. This formulation remains non-unique, but can be solved using methods such as

singular value decomposition, pseudo-inverse, and Moore–Penrose pseudo-inverse.

The topic of rate-independent single crystal plasticity is still under active discussion. Several concerns remain regarding algorithm stability and computational efficiency [11, 12, 42, 44].

Objectives and layout of this thesis. The main objective of this thesis is to review, implement and compare algorithms, based on those in Miehe and Schröder [29] and Schmidt-Baldassari [37], for rate-independent single-crystal plasticity. The starting point is the development of a mathematical model for single-crystal plasticity. This leads to the principle of maximum plastic dissipation which is cast into a constrained optimisation problem. We will discuss the general theory of constrained optimisation and apply these ideas to the mathematical model, which will result in several formulations. From the formulations we will develop algorithms to solve the problem. We will describe how one would develop the software, in an object orientated framework, for solving these problems. Finally, we will conduct numerical experiments to evaluate the level of agreement between the algorithms and compare performance.

This thesis comprises three parts. Part 1 consists of Chapters 2 and 3. The central topic of this part is to develop the theory of single-crystal plasticity. Chapter 2 provides a review of continuum mechanics, where we discuss the topics of kinematics, balance laws and elastic constitutive relations. Chapter 3 provides an overview of single-crystal plasticity. The approach taken first discusses classical polycrystalline plasticity in a general framework. We then focus on the physical theory of single-crystals which further specify the kinematics and flow relations. In order to complete the constitutive relations it is necessary to look into topic of constrained minimisation. Here we look into three methods: Lagrange multiplier, penalty and augmented Lagrangian methods. From these three approaches we develop three formulations of the flow rules.

Part 2 focuses on developing algorithms for the three sets of constitutive equations. This part is divided into four chapters, viz. Chapters 4 – 7. Chapter 4 focuses on some the numerical methods we shall be using to develop algorithms. These methods are the LU decomposition, the singular value decomposition, a globally convergent Newton–Raphson method, and a short section on fixed point mappings. In Chapter 5 we discuss the finite element method. In Chapter 6 we apply the numerical methods to the three formulations and develop algorithms. For the Lagrange multiplier method we develop two algorithms, so

in total we develop four algorithms. Chapter 7 describes how these have been implemented in an object-orientated paradigm. We discuss the finite element library used as well as suggest a class structure for the implementation. In the final part, Part 3, we conduct numerical examples. Here it is verified that the four algorithms produce results that agree with one another as well as with results in the literature. Results are produced for two experiments. The first experiment is intentionally constructed such that a uniform stress state is induced. This allows for an easy analysis of results at the quadrature point level. The second experiment is a comparatively more complex problem. The results are appropriate for analysis at a macroscopic level. This experiment is also appropriate for a quantitative analysis of the performance of the various algorithms.

Part I

A theoretical discussion of elasticity, classical plasticity and single crystal plasticity

Chapter 2

Background in continuum mechanics

In this section we set out a mathematical framework for continuum mechanics. Essential to the constitutive relations of plasticity is the classical theory of elasticity. For our purposes we only discuss linear elasticity as this will be required in crystal plasticity as this assumption is commonly made due to the dominance of the plastic deformation. This discussion is done in a standard manner by first considering the kinematics of a body (see [24, p. 66] and [18, p. 16] amongst numerous others). This leads to the introduction of strain measures. For our purposes a small strain assumption is sufficient. Next, we introduce the concept of stress (see [17, p. 66] amongst numerous others). Once this framework has been established we consider the balance of momentum and derive the equation of motion (see for example [18, p. 24]). This section is followed by a discussion of laws of thermodynamics (see for example [18, p. 33] and [9, p. 15]). Here an important result is the local dissipation inequality. This equation will be a focus of many future discussions.

We introduce the idea of elasticity (and some ideas of plasticity) by considering tension test results for a thin strip of material, see [24, p. 115]. This leads to a discussion of linear elasticity, which is approached from two standpoints, see [18, p. 28]. The first simply states the stress-strain relationship, but limited conclusions are drawn from this. The second approach assumes a form of the Helmholtz free energy. Then the results from thermodynamics are applied to the free energy function, from which further conclusions can

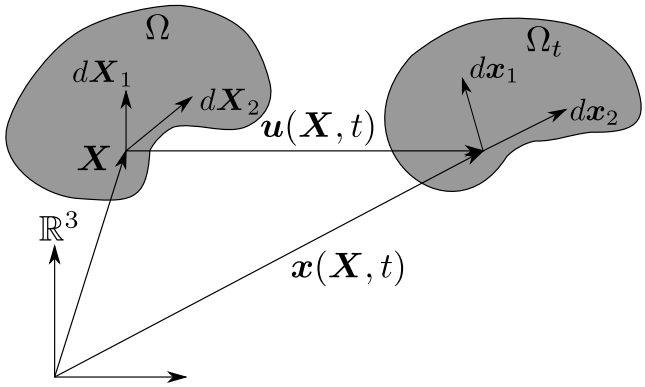


Figure 2.1: Current and reference configurations of an arbitrary material body. $d\boldsymbol{X}$ and $d\boldsymbol{x}$ are infinitesimal line elements in the reference and current configuration.

be made. This second approach stands as an example of how we will address the problem of plasticity.

2.1 Kinematics

We consider a body in its initial undeformed state which occupies a domain $\Omega \subset \mathbb{R}^3$. This state of the body is known as the *reference* configuration. The position vector of a material point of Ω is denoted by $\boldsymbol{X}$. The displaced state of the body at time t is denoted by Ω_t, and referred to as the *current* configuration, see Figure 2.1. The position at time t of the material point originally located at $\boldsymbol{X}$ is denoted by $\boldsymbol{x}$, such that

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{X}, t) \text{ with } \boldsymbol{x}(\boldsymbol{X}, 0) = \boldsymbol{X}.$$

We introduce a *displacement* vector $\boldsymbol{u}$, the norm of which is a measure of the distance between the reference and current configurations: that is,

$$\boldsymbol{u}(\boldsymbol{X}, t) = \boldsymbol{x}(\boldsymbol{X}, t) - \boldsymbol{X}. \tag{2.1}$$

In order to describe not only the displacement of a body, but also the deformation, additional quantities need to be introduced.

In order to describe deformation we consider infinitesimal line elements originating from some point within Ω, see Figure 2.1. We denote a line element as $d\boldsymbol{X}$. This line element

joins $\boldsymbol{X}$ to a neighbouring point $\boldsymbol{X} + d\boldsymbol{X}$.

We shall now consider the change of one line element after displacement has occurred. After displacement the point $\boldsymbol{X} + d\boldsymbol{X}$ has become $\boldsymbol{x} + d\boldsymbol{x}$. Using (2.1),

$$\begin{aligned} \boldsymbol{x} + d\boldsymbol{x} &= \boldsymbol{X} + d\boldsymbol{X} + \boldsymbol{u}(\boldsymbol{X} + d\boldsymbol{X}, t) \\ d\boldsymbol{x} &= d\boldsymbol{X} + \boldsymbol{u}(\boldsymbol{X} + d\boldsymbol{X}, t) + \boldsymbol{X} - \boldsymbol{x} \\ &= d\boldsymbol{X} + \boldsymbol{u}(\boldsymbol{X} + d\boldsymbol{X}, t) - \boldsymbol{u}(\boldsymbol{X}, t) \\ &= d\boldsymbol{X} + \nabla \boldsymbol{u} d\boldsymbol{X} \qquad \text{(to first order)} \\ &= (\boldsymbol{I} + \nabla \boldsymbol{u}) d\boldsymbol{X} \\ &= \boldsymbol{F} d\boldsymbol{X}. \end{aligned} \tag{2.2}$$

$\nabla \boldsymbol{u}$ is a second-order tensor known as the *displacement gradient*, with respect to the reference placement. The second-order tensor $\boldsymbol{F}$ is known as the *deformation gradient*, and maps a line element in the reference configuration to the current configuration. The Jacobian of this mapping is denoted as $J = \det(\boldsymbol{F}) > 0$.

The next important measure of deformation is derived by considering the relative change in two line elements with origin at the same point. Using (2.2), the two infinitesimal vectors $d\boldsymbol{X}_1$ and $d\boldsymbol{X}_2$ become

$$d\boldsymbol{x}_1 = d\boldsymbol{X}_1 + \nabla \boldsymbol{u} d\boldsymbol{X}_1, \quad d\boldsymbol{x}_2 = d\boldsymbol{X}_2 + \nabla \boldsymbol{u} d\boldsymbol{X}_2.$$

A measure of the relative change is given by the dot product of $d\boldsymbol{x}_1$ and $d\boldsymbol{x}_2$:

$$d\boldsymbol{x}_1 \cdot d\boldsymbol{x}_2 = d\boldsymbol{X}_1 \cdot d\boldsymbol{X}_2 + d\boldsymbol{X}_1 \cdot \nabla \boldsymbol{u} d\boldsymbol{X}_2 + d\boldsymbol{X}_2 \cdot \nabla \boldsymbol{u} d\boldsymbol{X}_1 + (\nabla \boldsymbol{u} d\boldsymbol{X}_1) \cdot (\nabla \boldsymbol{u} d\boldsymbol{X}_2).$$

Using the transpose on the last two terms we arrive at

$$d\boldsymbol{x}_1 \cdot d\boldsymbol{x}_2 = d\boldsymbol{X}_1 \cdot d\boldsymbol{X}_2 + d\boldsymbol{X}_1 \cdot [\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T + (\nabla \boldsymbol{u})^T \nabla \boldsymbol{u}] d\boldsymbol{X}_2. \tag{2.3}$$

We define the second-order tensor

$$\boldsymbol{\varepsilon}^* = \tfrac{1}{2}[\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T + (\nabla \boldsymbol{u})^T \nabla \boldsymbol{u}]. \tag{2.4}$$

It is clear from (2.3) that $\boldsymbol{\varepsilon}^*$ captures a measure of deformation in the neighbourhood of a point originally at $\boldsymbol{X}$. This measure is known as the *Lagrangian strain tensor*. If it is the case that the deformation of a body is small, in that the change in length of line elements $d\boldsymbol{x} \approx d\boldsymbol{X}$ or equivalently that $|\nabla \boldsymbol{u}| \ll 1$, then the product terms in $\boldsymbol{\varepsilon}^*$ are considered negligible and (2.4) becomes

$$\boldsymbol{\varepsilon}^* \approx \boldsymbol{\varepsilon} = \tfrac{1}{2}(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T) = \operatorname{sym}(\nabla \boldsymbol{u}).$$

This is known as the *small strain assumption*[1].

An important identity that allows one to transform areas in the current configuration to the reference configuration is *Nanson's formula*

$$\boldsymbol{n}da = J\boldsymbol{F}^{-T}\boldsymbol{N}dA,$$

where da is an area of a region in the current configuration, dA is the area in the reference configuration, and $\boldsymbol{n}$ is the outward normal to the area element in the current configuration while $\boldsymbol{N}$ is the outward normal in the reference configuration. Due to the effect of the small strain assumption on the deformation gradient one finds that Nanson's formula becomes

$$\boldsymbol{n}da \approx \boldsymbol{N}dA, \tag{2.5}$$

implying that the change in area is small.

We are also interested in the effect of the small strain assumption on the rate of change of an integral in the current configuration with an arbitrary integrand $\mathcal{F}$

$$\frac{d}{dt}\int_{\Omega_t} \mathcal{F}dx.$$

By use of the Jacobian, the integral can be transformed into one over the reference configuration:

$$\begin{aligned}\frac{d}{dt}\int_{\Omega_t} \mathcal{F}dx &= \frac{d}{dt}\int_{\Omega} \mathcal{F}JdX \\ &\approx \frac{d}{dt}\int_{\Omega} \mathcal{F}dX.\end{aligned}$$

[1]From this point on we make the small strain assumption.

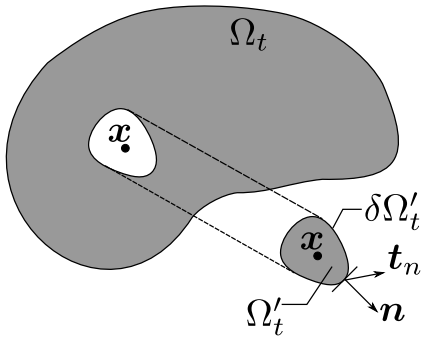


Figure 2.2: Euler's cut principle.

Since the reference configuration has no dependence on time, it implies that

$$\frac{d}{dt}\int_{\Omega_t} \mathcal{F} dx = \int_{\Omega} \dot{\mathcal{F}} dX. \tag{2.6}$$

This relationship will be useful later.

It can be shown that the change in the volume dV relative to the total volume V is given by

$$\frac{dV}{V} = \frac{\partial u_i}{\partial x_i} := \operatorname{div} \boldsymbol{u}.$$

Using the strain, this can be written as

$$\frac{dV}{V} = \operatorname{tr}(\boldsymbol{\varepsilon}). \tag{2.7}$$

2.2 Stress

We assume the existence of an internal stress within a body. One can imagine this as the force that an infinitesimal volume element within Ω_t applies to neighbouring volume elements. A first-order tensor quantity would not suffice to describe this. In order to elaborate on this we apply *Euler's cut principle*, see Figure 2.2. We cut a small region Ω'_t, with boundary $\partial\Omega'_t$, out of Ω_t. We apply a traction force to $\partial\Omega'_t$ such that the region is in the same state prior to its removal. This traction is represented by the vector $\boldsymbol{t}_n$, which is the traction on a surface with unit normal $\boldsymbol{n}$. If we multiply the traction by an infinitesimal

surface area da one gets the infinitesimal force

$$d\boldsymbol{f} = \boldsymbol{t}_n da$$

exerted on da. According to *Cauchy's theorem* [17, p. 71], the surface traction is a linear function of the surface normal $\boldsymbol{n}$,

$$\boldsymbol{t}_n := \boldsymbol{\sigma}\boldsymbol{n},$$

where the second-order tensor $\boldsymbol{\sigma}$ is the *Cauchy stress*, which relates the internal force of the cut region to the deformed area.

If one takes a small strain assumption we have that

$$\begin{aligned} d\boldsymbol{f} &= \boldsymbol{\sigma}\boldsymbol{n} da \\ &= \boldsymbol{\sigma}\boldsymbol{N} dA, \end{aligned} \tag{2.8}$$

due to the effect of the small strain assumption on Nanson's formula (2.5).

2.3 Balance of momentum

Next we would like to form a balance of linear and angular momentum. The rate of change of momentum of an arbitrary part Ω_t' of the body is equal to the total force applied to that part. These external forces can be classified into two sources. One of these is the *body force*, which has an effect per unit volume. A typical example of this is gravity. The second source is the *surface traction*, applied to the surface of the body. A mathematical statement of the balance of linear momentum is

$$\frac{d}{dt}\int_{\Omega_t'} \rho\dot{\boldsymbol{u}}\, dx = \int_{\Omega_t'} \rho\boldsymbol{b}\, dx + \int_{\partial\Omega_t'} \boldsymbol{t}\, ds,$$

where ρ is the density of the material, $\boldsymbol{b}$ is the body force and $\boldsymbol{t}$ is the surface traction experienced by the boundary of the part $\partial\Omega'_t$. Using the small strain assumption, by (2.6)

$$\begin{aligned}\int_{\Omega'} \rho\ddot{\boldsymbol{u}}\, dX &= \int_{\Omega'} \rho\boldsymbol{b}\, dX + \int_{\partial\Omega'} \boldsymbol{t}\, dS \\ &= \int_{\Omega'} \rho\boldsymbol{b}\, dX + \int_{\partial\Omega'} \boldsymbol{\sigma}\boldsymbol{N}\, dS. \end{aligned} \tag{2.9}$$

Using the divergence theorem on the third term of (2.9) we get

$$\int_{\Omega'} \left[\rho\ddot{\boldsymbol{u}} - \rho\boldsymbol{b} - \mathrm{div}\boldsymbol{\sigma}\right]\, dX = \boldsymbol{0}.$$

Since Ω' is an arbitrary part within Ω, the integrand must be zero. Thus we obtain the local form of the *equation of motion*

$$\mathrm{div}\boldsymbol{\sigma} + \rho\boldsymbol{b} = \rho\ddot{\boldsymbol{u}}. \tag{2.10}$$

In processes where the inertial effects are small, so-called *quasistatic* systems, the term $\ddot{\boldsymbol{u}} = \boldsymbol{0}$ and (2.10) becomes

$$\mathrm{div}\boldsymbol{\sigma} + \rho\boldsymbol{b} = \boldsymbol{0}. \tag{2.11}$$

One can also apply the principle of balance of angular momentum to our system. The balance of angular momentum states the the total moment acting on Ω'_t is equal to the rate of change of the angular momentum of Ω'_t. The mathematical definition is not given here, but after a lengthly derivation one concludes that the Cauchy stress is symmetric: that is,

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^T.$$

2.4 Boundary conditions and initial conditions for equation of motion

In order for the equation of motion (2.10) to be well posed we require boundary conditions on $\partial\Omega_t$ and inital conditions on Ω. We assume that $\partial\Omega_t$ can be decomposed into two sets;

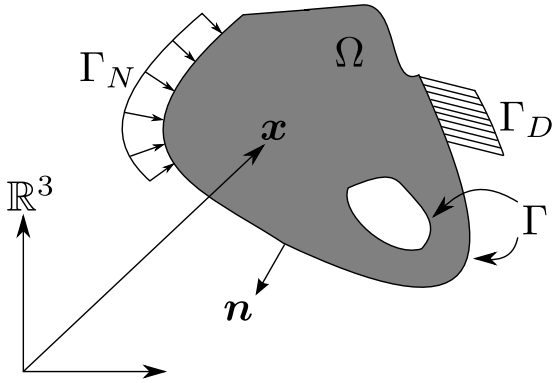


Figure 2.3: A schematic of the body Ω with applied boundary conditions.

Γ_D and Γ_N. The decomposition is such that

$$\partial\Omega_t = \overline{\Gamma_D \cup \Gamma_N}, \qquad \Gamma_D \cap \Gamma_N = \emptyset.$$

We use the over bar to indicate a set closure, for example:

$$\bar{\Omega} = \Omega \cup \partial\Omega.$$

Boundary conditions come in two forms, the *Dirichlet boundary condition* and the *Neumann boundary condition*. The Dirichlet boundary condition is in terms of displacement, that is

$$\boldsymbol{u} = \boldsymbol{g} \text{ on } \Gamma_D,$$

and Neumann boundary condition is in terms of an applied traction, that is

$$\boldsymbol{\sigma n} = \boldsymbol{h} \text{ on } \Gamma_N.$$

Here $\boldsymbol{n}$ is used to indicate the outward unit normal to $\partial\Omega_t$. For an illustration of the boundary conditions see Figure 2.3.

Initial conditions are prescribed conditions that describe the initial state of the body. The initial conditions required for (2.10) involve specificaion of both displacement and velocities;

that is

$$\boldsymbol{u}(\boldsymbol{X},0) = \boldsymbol{p}(\boldsymbol{X}) \text{ in } \Omega$$
$$\dot{\boldsymbol{u}}(\boldsymbol{X},0) = \boldsymbol{q}(\boldsymbol{X}) \text{ in } \Omega.$$

For the quasistatic system (2.11) inital conditions are not required.

2.5 Consequences of the laws of thermodynamics

For the sake of simplicity, i.e. restricting the discussion to plasticity, all processes are assumed to be isothermal. Regardless of this assumption, it is beneficial to approach the topic from a thermodynamic point of view, as it leads to several important conclusions. In what follows, the first and second laws of thermodynamics are applied to the problem.

For isothermal processes the first law of thermodynamics states that for any part Ω_t' of the body Ω_t, the rate of change of kinetic plus the internal energy is equal to the rate of work done on that part of the body by mechanical forces. The mathematical statement of this is

$$\frac{d}{dt}\int_{\Omega_t'} \left[\rho e + \tfrac{1}{2}\rho\dot{\boldsymbol{u}}^2\right]\, dx = \int_{\Omega_t'} \rho\dot{\boldsymbol{u}}\cdot\boldsymbol{b}\, dx + \int_{\partial\Omega_t'} \boldsymbol{t}_n\cdot\dot{\boldsymbol{u}}\, ds, \tag{2.12}$$

where e the internal energy per unit density. We use the divergence theorem to convert the third term into a volume integral; that is,

$$\begin{aligned}\int_{\partial\Omega_t'} \boldsymbol{t}_n\cdot\dot{\boldsymbol{u}}\, ds &= \int_{\partial\Omega_t'} \boldsymbol{\sigma}\boldsymbol{n}\cdot\dot{\boldsymbol{u}}\, ds \\ &= \int_{\Omega_t'} \boldsymbol{\sigma} : \nabla\dot{\boldsymbol{u}}\, dx + \int_{\Omega_t'} \mathrm{div}\boldsymbol{\sigma}\cdot\dot{\boldsymbol{u}}\, dx \\ &= \int_{\Omega_t'} \boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}}\, dx + \int_{\Omega_t'} \mathrm{div}\boldsymbol{\sigma}\cdot\dot{\boldsymbol{u}}\, dx.\end{aligned}$$

The last step is due to the symmetry of the stress. We use the notation $\dot{\boldsymbol{\varepsilon}} = \boldsymbol{\varepsilon}(\dot{\boldsymbol{u}})$. Using this result and (2.10), equation (2.12) simplifies to an expression for the rate of change of the internal energy in the form

$$\frac{d}{dt}\int_{\Omega_t'} \rho e\, dx = \int_{\Omega_t'} \boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}}\, dx.$$

By using (2.6) this becomes

$$\int_{\Omega'} \rho\dot{e}\, dX = \int_{\Omega'} \boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}}\, dX.$$

Since Ω' is arbitrary we can express this in the local form

$$\rho\dot{e} = \boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}}. \tag{2.13}$$

For isothermal processes the second law of thermodynamics states that the rate of increase in *entropy* η is non-negative. This leads to a mathematical statement

$$\frac{d}{dt}\int_{\Omega'_t} \eta\, dx \geq 0.$$

By (2.6) this becomes

$$\int_{\Omega'} \dot{\eta}\, dX \geq 0,$$

whose local statement is

$$\dot{\eta} \geq 0. \tag{2.14}$$

since Ω' is arbitrary.

The usual practice in continuum solid mechanics is to work with the *Helmholtz free energy* ψ, given by

$$\psi = \rho e - \eta\theta$$

where θ is the absolute temperature. Substituting this and (2.13) in (2.14), leads to the *local dissipation inequality* for constant θ

$$\dot{\psi} - \boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}} \leq 0. \tag{2.15}$$

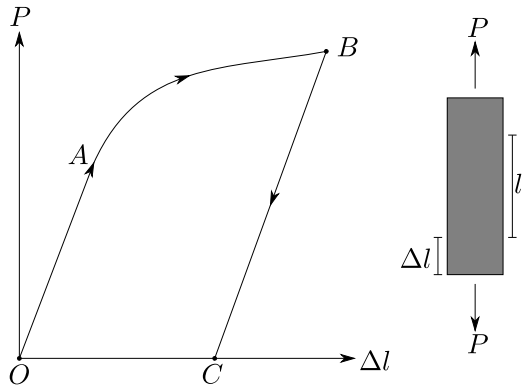


Figure 2.4: A strip in uniaxial tension.

2.6 General examples of material behaviour

Materials behave differently depending on their composition. To illustrate some of the different behaviours we consider a tension test on a thin strip of material. The strip has length l and the change in length, the elongation, is denoted by Δl. A uniaxial load P is applied. The relationship of applied load to elongation is shown in Figure 2.4.

It is observed that if the applied load ceases between OA the sample returns to its original length, i.e. $\Delta l = 0$. This behaviour within OA is called elasticity. In Figure 2.4 a linear relationship between applied load and elongation is depicted. This is known as *linear elasticity* and is discussed in a following section.

Applied loads greater than A result in *plastic* behaviour. As depicted in Figure 2.4, applied loads greater than A follow a path similar to $OABC$. There are several important properties of this type of behaviour. Firstly, once the external force is removed the elongation is not zero, i.e. at C, $P = 0$ and $\Delta l \neq 0$. Secondly, if one were to reload the strip from the point C the material would behave elastically to point B. In this case the slope of CB would be the same as OA. Finally, when loading from C plastic deformation will only occur for loads exceeding B.

Other types of behaviour beyond the point of initial yield, the point A, exist. This behaviour is called the *elostoplastic* behaviour. Some notable examples are *perfect plasticity* where AB has a zero slope, *hardening* where AB has a positive non-zero slope, and *softening* where AB has a negative non-zero slope.

2.7 Linear elasticity

Many materials experience deformations that are elastic. This behaviour can be described as a body returning to its initial configuration when external loads cease. The regime of this behaviour varies from material to material. In linear elasticity the stress is related linearly to the strain; that is,

$$\boldsymbol{\sigma} = \mathcal{C}^e \boldsymbol{\varepsilon},$$

or in index notation

$$\sigma_{ij} = \mathcal{C}^e_{ijkl} \varepsilon_{kl},$$

where $\mathcal{C}^e$ is a fourth-order tensor of elastic moduli. This equation can be thought of as a generalised form of Hooke's law. Due to the symmetries of $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ it can be shown that $\mathcal{C}^e$ has minor symmetries

$$\mathcal{C}^e_{ijkl} = \mathcal{C}^e_{jikl} = \mathcal{C}^e_{ijlk}. \tag{2.16}$$

An alternative, but equivalent, view on elasticity is that one assumes that the Helmholtz free energy is only functions of strain, that is

$$\psi = \psi(\boldsymbol{\varepsilon}).$$

The dissipation inequality (2.15) implies

$$\left(\frac{\partial \psi}{\partial \boldsymbol{\varepsilon}} - \boldsymbol{\sigma} \right) : \dot{\boldsymbol{\varepsilon}} \leq 0.$$

Since the dissipation inequality holds for all $\dot{\boldsymbol{\varepsilon}}$ it follows that

$$\boldsymbol{\sigma} = \frac{\partial \psi}{\partial \boldsymbol{\varepsilon}}.$$

If we assume that the free energy function has a quadratic dependence on the strain, so

that

$$\psi(\boldsymbol{\varepsilon}) = \tfrac{1}{2}\boldsymbol{\varepsilon} : \mathcal{C}\boldsymbol{\varepsilon}, \tag{2.17}$$

then the stress is given by[2]

$$\boldsymbol{\sigma} = \tfrac{1}{2}(\mathcal{C} + \mathcal{C}^T)\boldsymbol{\varepsilon} = \mathrm{sym}(\mathcal{C})\boldsymbol{\varepsilon}. \tag{2.18}$$

If in (2.17) we replace $\mathcal{C}$ with $\mathrm{sym}(\mathcal{C})$ the free energy of any state will remain unaffected as $\boldsymbol{\varepsilon}$ is symmetric. Thus, in (2.17) we are free to replace $\mathcal{C}$ with its symmetric part. If we do so, and regard $\mathcal{C}^e$ as having major symmetries ($\mathcal{C}^e_{ijkl} = \mathcal{C}^e_{klij}$), then (2.17) and (2.18) are given by

$$\psi(\boldsymbol{\varepsilon}) = \tfrac{1}{2}\boldsymbol{\varepsilon} : \mathcal{C}^e\boldsymbol{\varepsilon} \quad \text{and} \quad \boldsymbol{\sigma} = \mathcal{C}^e\boldsymbol{\varepsilon}.$$

The symetries of $\mathcal{C}^e$ can also be shown from (2.17) as:

$$\mathcal{C}^e = \frac{\partial^2\psi}{\partial\boldsymbol{\varepsilon}\partial\boldsymbol{\varepsilon}}.$$

In index form this is

$$\mathcal{C}^e_{ijkl} = \frac{\partial^2\psi}{\partial\boldsymbol{\varepsilon}_{ij}\partial\boldsymbol{\varepsilon}_{kl}} = \frac{\partial^2\psi}{\partial\boldsymbol{\varepsilon}_{kl}\partial\boldsymbol{\varepsilon}_{ij}} = \frac{\partial^2\psi}{\partial\boldsymbol{\varepsilon}_{ji}\partial\boldsymbol{\varepsilon}_{lk}}.$$

Thus, $\mathcal{C}^e$ exhibits major and minor symmetries. In summary $\mathcal{C}^e$ is required to have the following symmetries:

$$\mathcal{C}^e_{ijkl} = \mathcal{C}^e_{jikl} = \mathcal{C}^e_{ijlk} = \mathcal{C}^e_{klij}.$$

[2]For a fourth-order tensor $\mathcal{C}$, we define the transpose $(\mathcal{C}^T)_{ijkl} = \mathcal{C}_{klij}$ and the symetric operator $\mathrm{sym}(\mathcal{C}) = 1/2(\mathcal{C} + \mathcal{C}^T)$.

Chapter 3

Polycrystal plasticity and single-crystal plasticity

Having introduced a mathematical framework of mechanics and discussed a simple model for elasticity, we move on to plasticity. We shall first introduce a general framework for plasticity and set out the relationships that any plastic theory must satisfy. Once these are clear, we discuss crystal structures from a physical point of view. Here we wish to obtain a better understanding of how the microscopic structure of a material has a significant effect on the way in which materials plastically deform. Next we apply the observations to specify the relationships in a theory of plasticity for single crystals. A very important result of classical plasticity is the classical principle of maximum plastic dissipation, which results from thermodynamical considerations. This statement can be presented in alternative forms using results from the constrained minimisation of functions. We discuss the relevant results, and then apply these to the classical principle of maximum plastic dissipation. The result of each such treatment is a set of flow rules which can be used as constitutive equations.

3.1 Classical plasticity

3.1.1 Discussion of plastic deformation

We turn our attention back to Section 2.6 and the tension test described in Figure 2.4. In general every material that behaves plastically, initially behaves elastically. We call the regime within which a material behaves elastically, the *elastic domain*. As Figure 2.4 illustrates a material has a point of initial yield, that is at the point A. Once plastic deformation has occurred and the external forces cease, one notices that the point of yield has changed. This is illustrated by the elastic behaviour between points C and B. The elastic domain and the point of yield have changed and are dependent on the plastic *history* of the material.

The behaviour that occurs beyond the point of yield is known as elasto-plastic behaviour. This change in behaviour is due to an internal force (or stress like quantity) developing in the material as a result of movement of crystal defects. This internal responce is known as *hardening*, and it differs from material to material. Plastic behaviour can be classed as perfect plasticity, kinematic hardening, isotropic hardening, and softening, amongst others.

We introduce a new strain measure. This is the strain that has accumulated due to plastic deformation. To justify this, consider a point like C in Figure 2.4 where the applied load is zero and plastic deformation has already occurred. At this point the sample is in a stress-free state, but the strain is nonzero. This permanent strain is due to plastic deformation, and called the *plastic strain*.

Clearly the extent of the elastic domain, the point at which yield occurs and the amount of hardening depend on the plastic history. In the next sections we cast these ideas into a mathematical framework. We shall introduce a new internal variable that keeps track of the plastic history of a material point. From this variable one will be able to determine the extent of the hardening, the point at which yield will occur, and the size of the elastic domain.

3.1.2 Classical plasticity

As an earlier section suggested, we introduce a set of internal kinematic variables ξ_i which account for the plastic deformation history. These internal variables may be scalars or tensors. The theory of thermodynamics does not allow internal variables that are vectors (see references within [18, p. 48]). We represent the set of these internal variables collectively as $\boldsymbol{\xi}$.

We also introduce the *total strain* $\boldsymbol{\varepsilon}$ which comprises of an elastic part $\boldsymbol{\varepsilon}^e$, which is due to reversible deformations of the lattice, and a plastic part $\boldsymbol{\varepsilon}^p$ which arises from permanent deformations due to the flow of dislocations. The strain is assumed to be related to its elastic and plastic parts additively, so that

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^e + \boldsymbol{\varepsilon}^p. \tag{3.1}$$

This can be shown to be a physically feasible assumption for small deformations, see [18, p. 50] or [25]. One can show that the additive decomposition of the strain leads an additive decomposition in the Helmholtz free energy function

$$\psi(\boldsymbol{\varepsilon}, \boldsymbol{\xi}) = \psi^e(\boldsymbol{\varepsilon}^e) + \psi^p(\boldsymbol{\xi}) := \psi(\boldsymbol{\varepsilon}^e, \boldsymbol{\xi}), \tag{3.2}$$

see [18, p. 51].

The thermodynamical considerations from Section 2.5 apply. The important result from that section was the dissipation inequality (2.15). Substitution of (3.1) and (3.2) into (2.15) yields

$$-\left(\boldsymbol{\sigma} - \frac{\partial \psi^e}{\partial \boldsymbol{\varepsilon}^e}\right) : \dot{\boldsymbol{\varepsilon}}^e + \left(\frac{\partial \psi^p}{\partial \boldsymbol{\xi}} \cdot \dot{\boldsymbol{\xi}} - \boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}}^p\right) \leq 0. \tag{3.3}$$

Because this must hold for any elastic strain rate $\dot{\boldsymbol{\varepsilon}}^e$ it follows from a classical Coleman-Noll that

$$\boldsymbol{\sigma} - \frac{\partial \psi^e}{\partial \boldsymbol{\varepsilon}^e} = \mathbf{0} \quad \Rightarrow \quad \boldsymbol{\sigma} = \frac{\partial \psi^e}{\partial \boldsymbol{\varepsilon}^e}$$

and

$$\frac{\partial \psi^p}{\partial \boldsymbol{\xi}} \cdot \dot{\boldsymbol{\xi}} - \boldsymbol{\sigma} : \dot{\varepsilon}^p \leq 0. \tag{3.4}$$

The second of these is known as the *reduced dissipation inequality*. We define the force quantity conjugate to the internal variable $\boldsymbol{\xi}$ as

$$\boldsymbol{\chi} := -\frac{\partial \psi^p}{\partial \boldsymbol{\xi}}.$$

We identify the elements of $\boldsymbol{\chi}$ with hardening. Now (3.4) becomes

$$\boldsymbol{\chi} \cdot \dot{\boldsymbol{\xi}} + \boldsymbol{\sigma} : \dot{\varepsilon}^p \geq 0. \tag{3.5}$$

One can write this in a more concise manner by introducing the *generalised plastic strain* $\boldsymbol{Q} = \{\varepsilon^p, \boldsymbol{\xi}\}$ and the *generalised stress* $\boldsymbol{\Sigma} = \{\boldsymbol{\sigma}, \boldsymbol{\chi}\}$. The double contraction of these quantities is defined by

$$\boldsymbol{\Sigma} : \dot{\boldsymbol{Q}} = \boldsymbol{\sigma} : \dot{\varepsilon}^p + \boldsymbol{\chi} \cdot \dot{\boldsymbol{\xi}} \geq 0. \tag{3.6}$$

Next we consider the *set of admissible stresses*. One can see from Figure 2.4 that given a state of a material, there are only certain values that the stress can reach without further plastic deformation. There are also values which once reached, with the appropriate loading conditions, plastic deformation will begin to occur. We call this set the set of admissible stress $\mathcal{E}$, and its boundary $\mathcal{B}$ the yield surface. One can also see from Figure 2.4 that once plastic deformation has occurred this set of values has changed. This suggests that this set depends on the $\boldsymbol{\chi}$. This set and its boundary can be expressed in terms of a function ϕ; that is

$$\mathcal{B} = \{\boldsymbol{\Sigma} \mid \phi(\boldsymbol{\Sigma}) = 0\} \qquad \mathcal{E} = \{\boldsymbol{\Sigma} \mid \phi(\boldsymbol{\Sigma}) \leq 0\}.$$

Next we elaborate on the types of behaviour that occur. Plastic materials exhibit elastic behaviour under two loading conditions: firstly, when the applied load is not sufficient for the onset of plastic behaviour i.e. the generalised stress state is not on $\mathcal{B}$; and secondly, when plastic loading is occurring and the loading conditions are such that plastic deformation

ceases, i.e. the generalised stress state is on $\mathcal{B}$ but $\dot{\phi} < 0$. Plastic materials illustrate plastic behaviour only when on the yield surface $\mathcal{B}$. A mathematical statement of this is

$$\dot{\boldsymbol{Q}} = \mathbf{0} \text{ if } \begin{cases} \phi(\boldsymbol{\Sigma}) < 0 \\ \phi(\boldsymbol{\Sigma}) = 0 \text{ and } \dot{\phi} < 0, \end{cases}$$

$$\dot{\boldsymbol{Q}} \neq \mathbf{0} \text{ only if } \phi(\boldsymbol{\Sigma}) = 0 \text{ and } \dot{\phi} = 0.$$

These conditions will be shown to be the case in the following section.

We require a final postulate, which is the postulate of *maximum plastic work*. It can be justified by physical arguments, see [25]. This postulate states: given a state of generalised stress $\boldsymbol{\Sigma}$, for which $\phi(\boldsymbol{\Sigma}) = 0$ and a generalised plastic strain rate $\dot{\boldsymbol{Q}}$ associated with $\boldsymbol{\Sigma}$, then

$$\boldsymbol{\Sigma} : \dot{\boldsymbol{Q}} \geq \boldsymbol{T} : \dot{\boldsymbol{Q}}, \quad \forall \boldsymbol{T} \in \mathcal{E}. \tag{3.7}$$

The implication of this inequality will be discussed in the following section, but it is clear from (3.6) that we seek to maximise dissipation during plastic deformation.

If one uses (3.1) in (2.7) one can write

$$\frac{dV}{V} = \operatorname{tr}(\boldsymbol{\varepsilon}^e + \boldsymbol{\varepsilon}^p) = \operatorname{tr}(\boldsymbol{\varepsilon}^e) + \operatorname{tr}(\boldsymbol{\varepsilon}^p).$$

"In metal plasticity it is observed that volume changes are almost exclusively of an elastic nature" [18, p. 53]. So it follows that

$$\operatorname{tr}(\boldsymbol{\varepsilon}^p) = 0 \tag{3.8}$$

and

$$\frac{dV}{V} = \operatorname{tr}(\boldsymbol{\varepsilon}^e).$$

That is, plastic flow is incompresible.

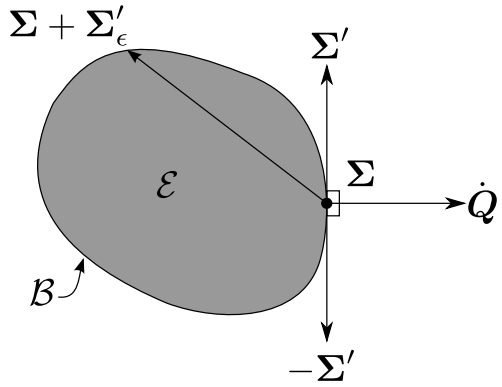


Figure 3.1: The set of admissible generalised stress used for illustrating the normality of the generalised strain.

3.1.3 Normality laws

In this section we follow the argument presented in [18], for smooth yield functions. Let $\boldsymbol{\Sigma} = \{\boldsymbol{\sigma}, \boldsymbol{\chi}\}$ be the set of generalised stresses and $\boldsymbol{Q} = \{\boldsymbol{\varepsilon}^p, \boldsymbol{\xi}\}$ the set of generalised plastic strains. We assume the existence of a yield function $\phi(\boldsymbol{\Sigma})$ from which a set of admissible generalised stress $\mathcal{E} = \{\boldsymbol{\Sigma} \mid \phi(\boldsymbol{\Sigma}) \leq 0\}$ and a yield surface $\mathcal{B} = \{\boldsymbol{\Sigma} \mid \phi(\boldsymbol{\Sigma}) = 0\}$ can be defined. A requirement on any yield function is that $\mathbf{0} \in \mathcal{E}$. The evolution of the internal variable $\boldsymbol{Q}$ is governed by the maximum plastic dissipation inequality (3.7). We shall now prove that $\boldsymbol{Q}$ lies in the direction of the normal to yield surface at $\boldsymbol{\Sigma}$. Figure 3.1 serves as an illustration for this proof. Let $\boldsymbol{\Sigma}'$ be a unit tangent vector lying in the tangent hyperplane of the yield surface at $\boldsymbol{\Sigma}$. Now consider a sequence of generalised stress $\boldsymbol{T} = \boldsymbol{\Sigma} + \boldsymbol{\Sigma}'_\epsilon$ that have the properties $\boldsymbol{\Sigma}'_\epsilon \to \mathbf{0}$ and

$$\frac{\boldsymbol{\Sigma}'_\epsilon}{||\boldsymbol{\Sigma}'_\epsilon||} \to \boldsymbol{\Sigma}' \quad \text{as} \quad \epsilon \to 0.$$

Using (3.7),

$$\begin{aligned} &\boldsymbol{\Sigma} : \dot{\boldsymbol{Q}} \geq \left(\boldsymbol{\Sigma} + \boldsymbol{\Sigma}'_\epsilon\right) : \dot{\boldsymbol{Q}} \\ \Rightarrow &\boldsymbol{\Sigma}'_\epsilon : \dot{\boldsymbol{Q}} \leq 0 \\ \Rightarrow &\frac{\boldsymbol{\Sigma}'_\epsilon}{||\boldsymbol{\Sigma}'_\epsilon||} : \dot{\boldsymbol{Q}} \leq 0. \end{aligned}$$

Taking the limit as $\epsilon \to 0$,

$$\boldsymbol{\Sigma}' : \dot{\boldsymbol{Q}} \leq 0. \tag{3.9}$$

This can be repeated for a similar sequence of generalised stresses with the property

$$\frac{\boldsymbol{\Sigma}'_\epsilon}{||\boldsymbol{\Sigma}'_\epsilon||} \to -\boldsymbol{\Sigma}' \quad \text{as} \quad \epsilon \to 0.$$

This will lead to

$$\boldsymbol{\Sigma}' : \dot{\boldsymbol{Q}} \geq 0. \tag{3.10}$$

It is clear from (3.9) and (3.10) that

$$\boldsymbol{\Sigma}' : \dot{\boldsymbol{Q}} = 0.$$

This is true for any tangent vector $\boldsymbol{\Sigma}'$ at $\boldsymbol{\Sigma}$. Thus $\dot{\boldsymbol{Q}}$ is normal to $\mathcal{B}$ at $\boldsymbol{\Sigma}$. This implies that when plastic flow occurs, $\dot{\boldsymbol{Q}}$ can be written as

$$\dot{\boldsymbol{Q}} = \dot{\lambda} \nabla_{\boldsymbol{\Sigma}} \phi(\boldsymbol{\Sigma}),$$

where the *plastic multiplier* $\dot{\lambda}$ (a rate) is a scalar.

A further conclusion about the direction of $\dot{\boldsymbol{Q}}$ can be made. Equation (3.7) holds for any $\boldsymbol{T} \in \mathcal{E}$. If we choose $\boldsymbol{T} = \boldsymbol{0}$ then

$$\begin{aligned} \boldsymbol{\Sigma} : \dot{\boldsymbol{Q}} &\geq 0, \\ \dot{\lambda}\boldsymbol{\Sigma} : \nabla_{\boldsymbol{\Sigma}} \phi(\boldsymbol{\Sigma}) &\geq 0. \end{aligned} \tag{3.11}$$

We can argue that because ϕ is convex and contains $\boldsymbol{\Sigma} = \boldsymbol{0}$, it implies that the "sign" of $\boldsymbol{\Sigma}$ and $\nabla_{\boldsymbol{\Sigma}} \phi$ are equal, thus $\boldsymbol{\Sigma} : \nabla_{\boldsymbol{\Sigma}} \phi(\boldsymbol{\Sigma}) \geq 0$. If we use this in (3.11), it follows that $\dot{\lambda} \geq 0$. This implies that not only is $\dot{\boldsymbol{Q}}$ normal to $\mathcal{B}$ at $\boldsymbol{\Sigma}$, but it is in the same direction as the outward normal of $\mathcal{B}$.

The expression for $\boldsymbol{Q}$ can be separated into the components corresponding to $\boldsymbol{\varepsilon}^p$ and ξ by

writing

$$\dot{\boldsymbol{\varepsilon}}^p = \dot{\lambda}\frac{\partial \phi}{\partial \boldsymbol{\sigma}} \quad \text{and} \quad \dot{\boldsymbol{\xi}} = \dot{\lambda}\frac{\partial \phi}{\partial \boldsymbol{\chi}}. \tag{3.12}$$

Plastic strain can only occur when on the yield surface. This gives rise to constraints on $\dot{\lambda}$ and ϕ known as the *Karush–Kuhn–Tucker* (KKT) conditions:

$$\dot{\lambda} \geq 0, \quad \phi \leq 0 \quad \text{and} \quad \dot{\lambda}\phi = 0.$$

This first condition constrains the sign of $\dot{\lambda}$. The second condition comes from the definition of the yield condition. The third condition implies that either $\dot{\lambda} \geq 0$ and $\phi = 0$ associated with plastic deformation, or $\dot{\lambda} = 0$ and $\phi \leq 0$ associated with no plastic deformation.

Consider a generalised stress state $\hat{\boldsymbol{\Sigma}}$ such that $\phi(\hat{\boldsymbol{\Sigma}}) = 0$. Due to the constraint $\phi \leq 0$, we must have $\dot{\phi}(\hat{\boldsymbol{\Sigma}}) \leq 0$. If $\dot{\phi}(\hat{\boldsymbol{\Sigma}}) < 0$ we are moving off the yield surface and re-entering the elastic domain. This is know as elastic unloading, and has associated $\dot{\lambda} = 0$. If $\dot{\phi}(\hat{\boldsymbol{\Sigma}}) = 0$ then we are staying on the yield surface, thus plastic deformation is occurring. This is called plastic loading, and has associated $\dot{\lambda} > 0$. This behaviour can be summarised by the conditions:

$$\text{when } \phi = 0, \text{ then } \dot{\lambda} \geq 0, \quad \dot{\phi} \leq 0 \quad \text{and} \quad \dot{\lambda}\dot{\phi} = 0.$$

The third condition is know as the *consistency condition*. When the KKT are taken into consideration, the consistency condition is valid not only at $\hat{\boldsymbol{\Sigma}}$ but rather for all $\boldsymbol{\Sigma}$.

3.2 Crystal plasticity

In this section the constitutive relations for single crystal plasticity are given. The physical microstructure is discussed in order to justify many of the concepts, for further reading on this topic see Barrett [8, p. 2]. Lastly, the appropriate use of microstructural information, usually given in terms of a unit cell, is demonstrated.

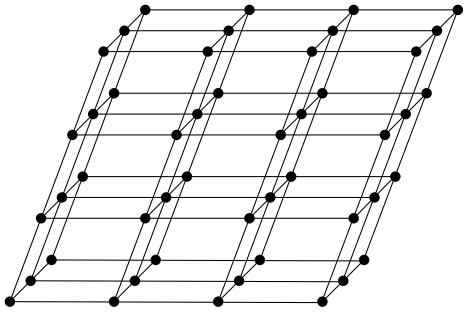


Figure 3.2: A space lattice and lattice points.

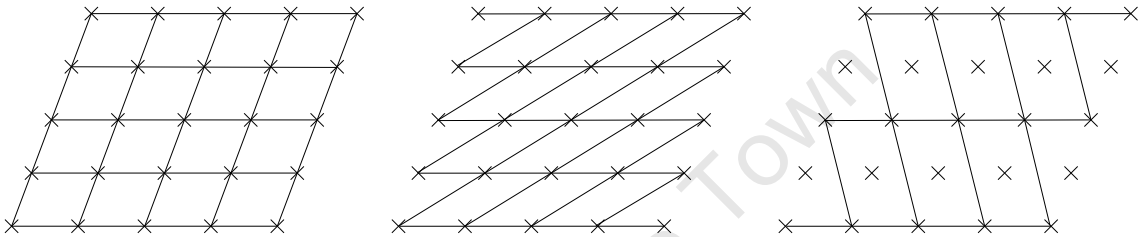


Figure 3.3: Non-uniqueness of the space lattice.

3.2.1 Crystal structures

At a microscopic level crystalline solids consist of atoms arranged in a pattern that repeats periodically in three dimensions. As we shall see, this microscopic ordering is very important in the plastic deformation of crystalline solids. In this section we briefly discuss this microscopic phenomenon.

In order to describe the crystal structure we introduce the *space lattice*. A representation of a space lattice is shown in Figure 3.2. It consists of straight lines connected together in a similar manner such that space is subdivided into equal prisms. Notice that the space is covered by the prisms with no voids or gaps. The points where the lines intersect are called *lattice points*, and are very important for the description of crystal structures. Atoms or clusters of atoms are placed at lattice points, depending on the material. Often in the case of metals, single atoms are placed at the lattice points. The lines drawn between lattice points are not unique; for this reason the space lattice is not unique (see Figure 3.3 for an illustration). It is also possible for a space lattice to have lattice points that are not at the vertices of the prisms (Figure 3.3(c)). These lattice points may lie at the centre of the

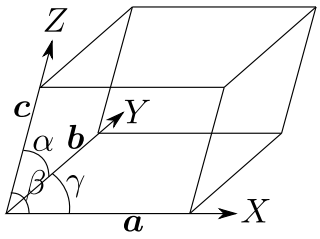


Figure 3.4: A space lattice's coordinate system.

prism, or at the centre of one of the faces of the prism. This is often done because it leads to a better illustrations of the possible behaviour.

An important idea to understand is that at every lattice point the surrounding lattice points have the same orientation. If an observer where placed at a lattice point and then moved to another lattice point, he would see the exact same structure and be unaware that he is at a new lattice point.

All possible arrangements of lattice points can be classified into one of 14 different spaces lattices. That is to say that there are no more than 14 ways that points can be arranged in space such that each point has identical surroundings. This does not imply that there are only 14 different crystal structures, but rather that the crystal structure consists of some fundamental pattern at each point.

One can define a coordinate system for a space lattice where one of the lattice points is chosen as the origin. Then the three vectors $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ that point to the closest vertex of the prism are chosen as the basis vectors. Here we shall refer to the axes described by these vectors as X, Y and Z. The vectors $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ have lengths a, b and c and are separated by angles α, β and γ (see Figure 3.4). The axes X, Y and Z are not necessarily orthogonal. In general the vectors fall into one of seven classes of *crystal system*. These are summarised in Figure 3.5.

Each space lattice can be broken down to its simplest building block, a parallelepiped called a *unit cell*. Each unit cell within a space lattice has exactly the same size and orientation. A unit cell that contains only lattice points at its vertices is called *primitive*. As previously mentioned, a unit cell can have lattice points at other points within them. See Figure 3.5 for all the unit cells associated with the 14 different lattice spaces.

It is necessary to develop a way to classify different planes that contain lattice points. We

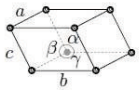
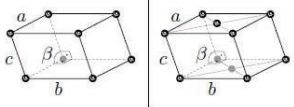
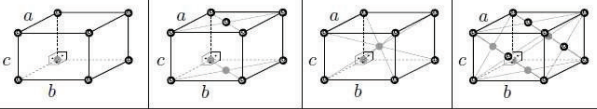
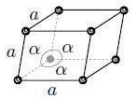
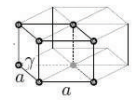
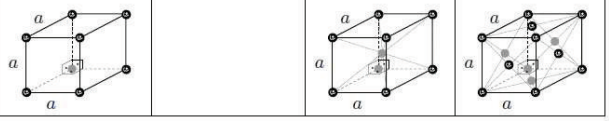
Symmetry	Primitive	C-centered	B-centered	F-centered
triclinic: a, b, c arbitrary, α, β, γ arbitrary				
monoclinic: a, b, c arbitrary, $\alpha = \gamma = \frac{\pi}{2} \neq \beta$				
orthorhombic: a, b, c arbitrary, $\alpha = \beta = \gamma = \frac{\pi}{2}$				
tetragonal: $a = b \neq c$, $\alpha = \beta = \gamma = \frac{\pi}{2}$				
trigonal: $a = b = c$, $\alpha = \beta = \gamma \neq \frac{\pi}{2}$				
hexagonal: $a = b \neq c$, $\alpha = \beta = \frac{\pi}{2} = \frac{1}{3}\gamma$				
cubic: $a = b = c$, $\alpha = \beta = \gamma = \frac{\pi}{2}$				

Figure 3.5: The seven classes of crystal system and the associated 14 unit cells [9].

are more interested in the orientation of the plane than its position. A plane is classified by its *Miller indices*. Every plane of lattice points intersects the XYZ coordinate system. The plane $A'B'C'$ has intersects $\boldsymbol{a}' = m\boldsymbol{a}$, $\boldsymbol{b}' = n\boldsymbol{b}$ and $\boldsymbol{c}' = p\boldsymbol{c}$; where $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ are the basis vectors for the space lattice. The Miller index $(h\ k\ l)$ is computed from the intercepts by taking the reciprocals of the coefficients and then multiplying them by a common denominator

$$(h\ k\ l) = \left(\frac{N}{n}\ \frac{N}{m}\ \frac{N}{p}\right).$$

N is chosen such that h, k and l are the smallest possible integers. A plane that intersects on the negative side of a axis (say the X axis) is denoted by using an over bar, i.e. $(\bar{h}\ k\ l)$. If a plane is parallel to the X axis it is taken to have the intercept $n = \infty$, thus $h = 0$. This also applies to planes parallel to the Y and Z axes.

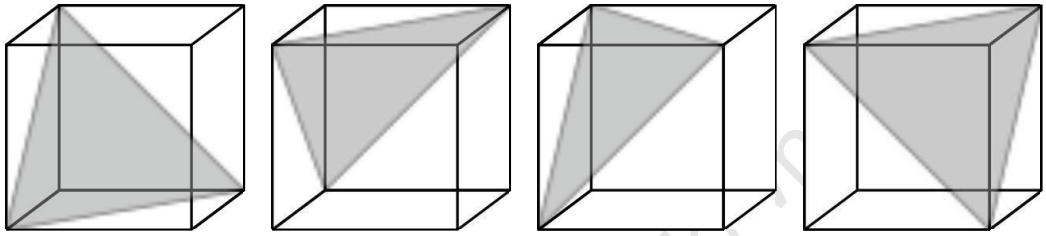
One can also define a family of planes that are equivalent in a crystal. For instance the planes (1 0 0), (0 1 0), (0 0 1), ($\bar{1}$ 0 0), (0 $\bar{1}$ 0) and (0 0 $\bar{1}$) form a family of planes because they are identical (apart from their orientation). A family of planes is indicated by curly braces, here that is {1 0 0}.

We also wish to construct an index system that gives the directions from one lattice point to another. Suppose the desired motion can be achieved by going u times $\boldsymbol{a}$, v times $\boldsymbol{b}$ and w times $\boldsymbol{c}$. If u, v and w are the smallest integers that could be used to describe the direction, then the direction is indicated as $[u\ v\ w]$. If one of the integers is negative this is indicated by an over bar, e.g. $[\bar{u}\ v\ w]$. An example of this is a direction along the X axis, given by [1 0 0].

At this point one may ask what this has to do with crystal plasticity. "Plastic deformation in a crystal occurs by movement of laminae of the crystal over one another" [8, p. 288]. This movement occurs along a crystallographic plane called the *slip plane* and in a direction called the *slip direction*. A given space lattice, at a given temperature and consisting of a given compound, has predefined planes along which slip may occur. The preferred plane of slip is highly dependent on the compound composition and temperature. A good example is a metal with a face-centered cubic (f.c.c.) unit cell at room temperature. For these materials slip usually happens in the plane that is most densely packed {1 1 1}. At higher temperatures other planes can become active. Another example is a metal with a body-

α	$\boldsymbol{s}^\alpha$	$\boldsymbol{m}^\alpha$	α	$\boldsymbol{s}^\alpha$	$\boldsymbol{m}^\alpha$	α	$\boldsymbol{s}^\alpha$	$\boldsymbol{m}^\alpha$
1	$[0\ 1\ \bar{1}]$	$(1\ 1\ 1)$	5	$[1\ 0\ 1]$	$(\bar{1}\ 1\ 1)$	9	$[\bar{1}\ \bar{1}\ 0]$	$(\bar{1}\ 1\ \bar{1})$
2	$[\bar{1}\ 0\ 1]$	$(1\ 1\ 1)$	6	$[\bar{1}\ \bar{1}\ 0]$	$(\bar{1}\ 1\ 1)$	10	$[0\ 1\ 1]$	$(1\ 1\ \bar{1})$
3	$[1\ \bar{1}\ 0]$	$(1\ 1\ 1)$	7	$[0\ 1\ 1]$	$(\bar{1}\ 1\ \bar{1})$	11	$[1\ \bar{1}\ 0]$	$(1\ 1\ \bar{1})$
4	$[0\ 1\ \bar{1}]$	$(\bar{1}\ 1\ 1)$	8	$[1\ 0\ \bar{1}]$	$(\bar{1}\ 1\ \bar{1})$	12	$[1\ 0\ 1]$	$(1\ 1\ \bar{1})$

Table 3.1: The slip planes and slip directions of the face-centered cubic unit cell, [19, p. 54].



(a) $(1\ 1\ 1)$ (b) $(1\ 1\ \bar{1})$ (c) $(\bar{1}\ 1\ \bar{1})$ (d) $(\bar{1}\ \bar{1}\ \bar{1})$

Figure 3.6: A representation of four of the slip planes of a metal with a f.c.c. unit cell at room temperature.

centered cubic (b.c.c.) unit cell at temperatures higher than room temperature. For these materials slip usually happens on the planes $\{1\ 1\ 0\}$, $\{1\ 2\ 3\}$ and $\{1\ 1\ 2\}$. The slip directions show strong dependence on the type of unit cell only, not the compound or the temperature. For a given slip plane, the slip direction is always the direction of the closest packed row of lattice points within that plane. A slip plane may have more than one slip direction because the rows are equivalently dense. The slip plane, along with a slip direction within the plane, is known as a *slip system*. This has great implications for plasticity, because now the direction of the plastic deformation is confined in specific planes along specific directions. A good example is that of metals with a f.c.c. unit cell at room temperature. This material has 12 slip systems, given in Table 3.1; a visual representation of some of the the slip planes is given in Figure 3.6. In Table 3.1 the slip dirrection and slip plane normal of the α^{th} slip system is denoted $\boldsymbol{s}^\alpha$ and $\boldsymbol{m}^\alpha$, respectively. Metals with a b.c.c. unit cell at temperatures higher than room temperature have 48 slip systems.

Analogous to classical plasticity, slip systems also have a yield stress and experience hardening. A slip system will only yield and begin to slip when the magnitude of the *critical resolved shear stress* is reached. The resolved shear stress is the magnitude of the stress along the slip direction on the slip plane. The value of the critical resolved shear stress can

change as a result of prior deformation of the crystal. Thus, hardening occurs on the slip plane itself. In crystalline structures the phenomenon of *latent hardening* or *cross hardening* refers to an increase in the critical resolved shear stress of a slip system induced by slip on other slip systems.

For stresses below those that produce slip, a crystal experiences elastic deformations. These deformations are associated with a distortion of the crystal structure or space lattice. The distortions are small, and as a result a linear elastic relationship between the stress and the strain is acceptable for deformation in the elastic range. Due to the underlying order of the crystal structure, the elastic response differs depending on the orientation crystal structure relative to the axis of applied load. For this reason the elastic response is considered to be anisotropic.

Many engineering materials have a polycrystalline structure. This includes most common metals and ceramics. This means that the material consists of *grains*. Each of the grains has the same ordering, or unit cell, but the orientation of the unit cell differs from grain to grain. A *single crystal* is a material that contains a single grain, which means the ordering can be described entirely by one unit cell.

3.2.2 Single crystal plasticity

We intend to incorporate many of the ideas in Section 3.2.1 into a mathematical framework. Some of the fundamental ideas from that discussion are:

- A single crystal is a material which, at microscopic level, has a perfectly ordered crystal structure. This microstructure can be described by a unit cell.
- Depending on the type of compound and unit cell (we assume a constant temperature), there are preferred slip planes.
- There are also preferred directions on these planes in which plastic slip will occur.
- A pair of slip plane and slip direction is called a slip system.
- Slip is directly linked to the plastic deformation of the material.
- A slip system will begin to experience slip when a critical resolved shear stress is reached.

- A slip system also experiences strain hardening as a result of prior slip.
- The crystal structure behaves elastically for stresses less than the critical resolved shear stress.
- This elastic deformation is anisotropic in nature.

Assume that a material has N slip systems. For a given slip plane and direction we define two slip systems for each plane normal, but with opposing directions. This means we can number the N slip systems in an ordered fashion $\alpha = 1, ..., 2N$. The α^{th} slip system's slip plane and slip direction are denoted $\boldsymbol{m}^\alpha$ and $\boldsymbol{s}^\alpha$. The reason why we have $2N$ slip systems is because we have doubly defined the slip systems. We illustrate this by example; if slip is experienced on the plane with Miller index (1 0 0) and along the direction [1 0 0], then we define two slip systems both with Miller index (1 0 0), but one with slip direction [1 0 0] and the other with slip direction [$\bar{1}$ 0 0]. The reason for this is because it affects the choice of yield function (this will be discussed later). It will also lead to a situation where the plastic multipiers can only be positive, which has parallels with classical plasticity. For convenience, we define the geometric tensor of the α^{th} slip system as

$$\boldsymbol{P}^\alpha = \text{sym}(\boldsymbol{s}^\alpha \otimes \boldsymbol{m}^\alpha).$$

We define two sets; *active* set $\mathcal{A}$, and the *potentially active* set $\mathcal{P}$. The active set is the set of slip systems that are experiencing slip at a given time. The potentially active set is the set of all slip systems $\mathcal{P} = \{\alpha \mid \alpha = 1, ..., 2N\}$.

As we have said, slip (or yield) will only occur when the critical resolved shear stress is reached. The resolved shear stress is the magnitude of the stress along the slip direction on the slip plane. The resolved shear stress is also called the *Schmid stress*. It is not an unreasonable assumption that the Schmid stress will enter the yield function for a slip system. Using Cauchy's formula (2.8), one can find the traction caused by the stress on the plane described by the normal $\boldsymbol{m}^\alpha$; that is $\boldsymbol{t}_m = \boldsymbol{\sigma}\boldsymbol{m}^\alpha$. We now orthogonally project the traction vector onto $\boldsymbol{s}^\alpha$ to find the magnitude of the projection. Thus the Schmid stress τ^α is given by

$$\tau^\alpha = (\boldsymbol{\sigma}\boldsymbol{m}^\alpha) \cdot \boldsymbol{s}^\alpha.$$

In index notation this is

$$\tau^\alpha = \sigma_{ij} s_i^\alpha m_j^\alpha.$$

This can also be writen as

$$\tau^\alpha = \boldsymbol{\sigma} : \boldsymbol{P}^\alpha.$$

We will introduce a new internal variable ξ^α to capture the accumulated amount of slip that has occurred on slip system α. In contrast to classical plasticity this variable need only be a scalar as we know the direction of plastic flow but not its magnitude. We represent the set of these internal variables collectively as $\boldsymbol{\xi} \in \mathbb{R}^{2N}$. The amount of elastic and plastic deformation the crystal structure has undergone is represented by the *elastic strain* $\boldsymbol{\varepsilon}^e$ and *plastic strain* $\boldsymbol{\varepsilon}^p$, respectively. Thus the *generalised strain* is

$$\boldsymbol{E} = \{\boldsymbol{\varepsilon}^e, \boldsymbol{\varepsilon}^p, \boldsymbol{\xi}\}.$$

We define the *generalised plastic strain*

$$\boldsymbol{Q} = \{\boldsymbol{\varepsilon}^p, \boldsymbol{\xi}\}.$$

As in Section 3.1.2 we use the additive decomposition assumption for the strain (3.1)

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^e + \boldsymbol{\varepsilon}^p, \tag{3.13}$$

and a small strain assumption

$$\boldsymbol{\varepsilon} = \mathrm{sym}(\nabla \boldsymbol{u}).$$

As discussed this leads to an additive decomposition of the Helmholtz free energy

$$\psi(\boldsymbol{\varepsilon}^e, \boldsymbol{\xi}) = \psi^e(\boldsymbol{\varepsilon}^e) + \psi^h(\boldsymbol{\xi}).$$

As previously mentioned the elastic deformations of a crystalline material are due to small distortion of the crystal structure. Since these deformations are small, it follows that an

appropriate assumption is that ψ^e has a quadratic dependence on the elastic strain,

$$\psi^e\left(\boldsymbol{\varepsilon}^e\right) = \tfrac{1}{2}\boldsymbol{\varepsilon}^e : \mathcal{C}^e\boldsymbol{\varepsilon}^e.$$

This leads, in the absence of plastic deformation, to a linear relationship between the stress and the strain, i.e. linear elastic material,

$$\boldsymbol{\sigma} = \frac{\partial\psi}{\partial\boldsymbol{\varepsilon}^e} = \frac{\partial\psi^e}{\partial\boldsymbol{\varepsilon}^e} = \mathcal{C}^e\boldsymbol{\varepsilon}^e. \tag{3.14}$$

Through the use of (3.13), (3.14) can be writen as

$$\boldsymbol{\sigma} = \mathcal{C}^e(\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^p). \tag{3.15}$$

The choice of the plastic contribution to the free energy depends on the type of hardening exhibited by the material. The material may exhibit kinematic hardening, isotropic hardening, or a perfectly plastic behaviour. For each of these cases a different form for ψ^h is chosen. In crystal plasticity the hardening rule is often not given as an explicit function, but rather it is defined via an evolution equation of the form

$$\dot{\chi}^\alpha(A, \dot{\xi}^\beta) = \sum_\beta h^{\alpha\beta}(A)\dot{\xi}^\beta, \tag{3.16}$$

where $A = \sum_\alpha \xi^\alpha$ is the *equivalent plastic strain* and $h^{\alpha\beta}$ a component of the matrix of *hardening moduli*. There are many proposed forms of hardening moduli, for example isotropic hardening $h^{\alpha\beta}(A) = h(A)$ proposed by Taylor [41] and independent hardening $h^{\alpha\beta}(A) = h(A)\delta^{\alpha\beta}$ proposed by Koiter [22] to name a few. Here we choose the latent hardening law propossed by Hutchinson [21] and Asaro [5],

$$h^{\alpha\beta}(A) = h(A)[q + (1-q)\delta^{\alpha\beta}]$$

where the function $h(A)$ is chosen according to type of hardening that is exhibited and $q \in [1,\ 1.4]$ is a parameter. The introduction of the additional parameter q allows for latent hardening. The following expression of $h(A)$, as well as the types of hardening they result

in, are relavent

$$h(A) = \begin{cases} h_0 \cosh^{-2}\left(\frac{h_0 A}{\tau_\infty - \tau_0^\alpha}\right) & \text{sech hardening} \\ h_1 & \text{linear hardening} \\ h_0 \cosh^{-2}\left(\frac{h_0 A}{\tau_\infty - \tau_0^\alpha}\right) + h_1 & \text{sech+linear hardening} \\ 0 & \text{perfect plastic.} \end{cases} \tag{3.17}$$

The set of admissible generalised stresses $\boldsymbol{\Sigma}$ is defined as the non-smooth convex set

$$\mathcal{E} = \{\boldsymbol{\Sigma} \mid \phi^\alpha(\boldsymbol{\Sigma}) \leq 0, \ \forall \ \alpha \in \mathcal{P}\},$$

where ϕ^α is the yield function associated with slip system α. It is defined by

$$\phi^\alpha(\boldsymbol{\sigma}, \chi^\alpha) = \tau^\alpha(\boldsymbol{\sigma}) - (\tau_0^\alpha - \chi^\alpha) \ \forall \ \alpha \in \mathcal{P}, \tag{3.18}$$

where $\tau^\alpha := \boldsymbol{\sigma} : \boldsymbol{P}^\alpha$ is the resolved Schmid stress and τ_0^α is the critical resolved shear stress. The set of admissible generalised stresses places a non-positive restriction on the yield function, that is $\phi^\alpha \leq 0$.

As suggested earlier, if we do not doubly define the slip systems then we use the yield function

$$\phi^\alpha(\boldsymbol{\sigma}, \chi^\alpha) = |\tau^\alpha(\boldsymbol{\sigma})| - (\tau_0^\alpha - \chi^\alpha) \ \forall \ \alpha \in \mathcal{Q},$$

where $\mathcal{Q} = \{\alpha | \alpha = 1, ..., m\}$. One can see from the normality conditions (3.12) that the modulus sign introduces its own complications. Doubly defining slip systems also introduces its own set of problems, in that this treatment has twice the number of equations to be solved.

The evolution of the internal variable $\boldsymbol{Q}$ is governed by the maximum plastic dissipation inequality (3.7). From this we get the normality, complementarity and consistency conditions; see section 3.1.3. The normality conditions are given by

$$\dot{\boldsymbol{\varepsilon}}^p = \sum_\alpha \dot{\lambda}^\alpha \frac{\partial \phi^\alpha}{\partial \boldsymbol{\sigma}} = \sum_\alpha \dot{\lambda}^\alpha \boldsymbol{P}^\alpha \quad \text{and} \quad \dot{\xi}^\alpha = \dot{\lambda}^\alpha \frac{\partial \phi^\alpha}{\partial \chi^\alpha} = \dot{\lambda}^\alpha, \tag{3.19}$$

where $\dot{\lambda}^\alpha$ is the plastic multiplier or, equivalently, the slip rate. The KKT conditions state that

$$\dot{\lambda}^\alpha \geq 0,\ \phi^\alpha \leq 0 \text{ and } \dot{\lambda}^\alpha \phi^\alpha = 0, \quad \alpha \in \mathcal{P} \tag{3.20}$$

and the consistency condition states that

$$\dot{\lambda}^\alpha \dot{\phi}^\alpha = 0.$$

In order for (3.19) to be well posed, they require a initial condition. We choose this to be $\boldsymbol{\varepsilon}^p(0, \boldsymbol{X}) = \mathbf{0}$.

An equivalent statement of the maximum plastic dissipation inequality (3.7) is the classical principle of maximum plastic dissipation: the plastic dissipation $\mathcal{D}^p \equiv \boldsymbol{\Sigma} : \dot{\boldsymbol{Q}}$ is maximised for a fixed $\dot{\boldsymbol{Q}}$; for all admissible generalised stresses

$$\boldsymbol{\Sigma} = \underset{\boldsymbol{T} \in \mathcal{E}}{\arg\max} \left[\boldsymbol{T} : \dot{\boldsymbol{Q}} \right] . \tag{3.21}$$

This statement can be approached from a *nonlinear programing* point of view. The following section gives a brief overview of this topic before we apply the principles to the problem.

The variables and equations relevant to crystal plasticity are summarised in Table 3.2.

3.2.3 A mathematical description of the unit cell

We have already mentioned many of the properties of the unit cell. Here we will describe a general unit cell in a mathematical framework. The unit cell is described by two objects: the slip systems and the constitutive tensor. The slip systems give us information that is useful when plastic deformation is occurring. The constitutive tensor describes how the crystal structure behaves under elastic deformations. As we have mentioned, the elastic deformations are of an anisotropic nature. This implies that the constitutive tensor that describes these deformations is required to be anisotropic. We define the unit cell basis $\hat{\boldsymbol{e}}_i$ and the fixed Cartesian basis $\boldsymbol{e}_i$. The unit cell information is defined in terms of the unit

Kinematic variables:	
displacement	$\boldsymbol{u}$
strain	$\boldsymbol{\varepsilon} = \text{sym}(\nabla \boldsymbol{u})$
elastic strain	$\boldsymbol{\varepsilon}^e = \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^p$
internal variables	$\lambda^1, \lambda^2, \ldots, \lambda^{2N}$
generalised plastic strain	$\boldsymbol{Q} = (\boldsymbol{\varepsilon}^p, \lambda^1, \lambda^2, \ldots, \lambda^{2N})$
Dynamic variables:	
stress	$\boldsymbol{\sigma}$
hardening	$\chi^1, \chi^2, ..., \chi^{2N}$
generalised stress	$\boldsymbol{\Sigma} = (\boldsymbol{\sigma}, \chi^1, \chi^2, ..., \chi^{2N})$
Scalar equations:	
free energy	$\psi(\boldsymbol{\varepsilon}, \lambda^\alpha) = \psi(\boldsymbol{\varepsilon}^e, \lambda^\alpha) = \psi^e(\boldsymbol{\varepsilon}^e) + \psi^p(\lambda^\alpha)$
elastic free energy	$\psi^e(\boldsymbol{\varepsilon}^e) = \frac{1}{2}\boldsymbol{\varepsilon}^e : \mathcal{C}^e \boldsymbol{\varepsilon}^e$
yield function	$\phi^\alpha(\boldsymbol{\sigma}, \chi^\alpha) = \boldsymbol{\sigma} : \boldsymbol{P}^\alpha - (\tau_0^\alpha - \chi^\alpha) \leq 0$
Equilibrium equation:	
	$\text{div}\, \boldsymbol{\sigma} + \rho \boldsymbol{b} = 0$
boundary conditions	$\boldsymbol{u} = \boldsymbol{g}$ on Γ_D and $\boldsymbol{\sigma n} = \boldsymbol{h}$ on Γ_N
Constitutive equations:	
stress	$\boldsymbol{\sigma} = \partial \psi^e / \partial \boldsymbol{\varepsilon}^e = \mathcal{C}^e \boldsymbol{\varepsilon}^e$
equivilent plastic strain	$A = \sum_\alpha \lambda^\alpha$
hardening rate	$\dot{\chi}^\alpha(A, \dot{\lambda}^\alpha) = \sum_\beta h^{\alpha\beta}(A) \dot{\lambda}^\beta$
	$h^{\alpha\beta}(A) = h(A)[q + (1-q)\delta^{\alpha\beta}]$
	e.g. $h(A) = h_0 \cosh^{-2}\left(\frac{h_0 A}{\tau_\infty - \tau_0^\alpha}\right)$
normality conditions	$\dot{\boldsymbol{\varepsilon}}^p = \sum_\alpha \dot{\lambda}^\alpha \boldsymbol{P}^\alpha$, $\boldsymbol{\varepsilon}^p(0, X) = \boldsymbol{0}$
complementarity conditions	$\dot{\lambda}^\alpha \geq 0$, $\phi^\alpha \leq 0$ and $\dot{\lambda}^\alpha \phi^\alpha = 0$
consistency condition	$\dot{\lambda}^\alpha \dot{\phi}^\alpha = 0$

Table 3.2: Summary of crystal plasticity equations.

cell basis:

$$\begin{aligned}\boldsymbol{s}^\alpha &= \hat{s}_i^\alpha \hat{\boldsymbol{e}}_i = s_i^\alpha \boldsymbol{e}_i \\ \boldsymbol{m}^\alpha &= \hat{m}_i^\alpha \hat{\boldsymbol{e}}_i = m_i^\alpha \boldsymbol{e}_i \\ \mathcal{C}^e &= \hat{\mathcal{C}}^e_{ijkl} \hat{\boldsymbol{e}}_i \otimes \hat{\boldsymbol{e}}_j \otimes \hat{\boldsymbol{e}}_k \otimes \hat{\boldsymbol{e}}_l = \mathcal{C}^e_{ijkl} \boldsymbol{e}_i \otimes \boldsymbol{e}_j \otimes \boldsymbol{e}_k \otimes \boldsymbol{e}_l.\end{aligned}$$

The slip system information is given in terms of the Miller indices and crystal structure directions. These are easily converted to vector form. A general Miller index $(u\ v\ w)$ is converted to the vector form by

$$(u\ v\ w) \ \Rightarrow\ \boldsymbol{m}^\alpha = \hat{m}_i^\alpha \hat{\boldsymbol{e}}_i,\ \hat{\boldsymbol{m}}^\alpha = \frac{1}{\sqrt{u^2+v^2+w^2}} \{u, v, w\}.$$

Similarly, a general crystal structure direction $[u\ v\ w]$ is converted to the vector form by

$$[u\ v\ w] \ \Rightarrow\ \boldsymbol{s}^\alpha = \hat{s}_i^\alpha \hat{\boldsymbol{e}}_i,\ \hat{\boldsymbol{s}}^\alpha = \frac{1}{\sqrt{u^2+v^2+w^2}} \{u, v, w\}.$$

For our purposes, we shall use either an isotopic or an anisotropic constitutive tensor. An isotropic tensor can be described by a set of two scalars. Here we choose λ and μ, Lamé's first parameter and shear modulus, respectively. The expression for an isotropic constitutive tensor is given by

$$\hat{\mathcal{C}}^e_{ijkl} = \lambda \delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}).$$

A conversion table is given in Table 3.3 to convert from the material parameters of κ, E and ν (the bulk modulus, Young's modulus and Poisson's ratio, respectively) to those of λ and μ.

An anisotropic constitutive tensor with cubic symmetries can be described by an additional set of three scalars, C_{11}, C_{12}, C_{44}. The anisotropic constitutive tensor with cubic symmetries is given by [14]:

$$\hat{\mathcal{C}}^e_{ijkl} = C_{12}\delta_{ij}\delta_{kl} + C_{44}(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) + (C_{11} - C_{22} - 2C_{44}) \sum_{r=1}^{3} \delta_{ir}\delta_{jr}\delta_{kr}\delta_{lr}. \tag{3.22}$$

We wish to represent the unit cell at various orientations. The problem is most easily

	λ	μ
$(E,\ \mu)$	$\frac{\mu(E-2\mu)}{3\mu-E}$	
$(\kappa,\ \lambda)$		$\frac{3(\kappa-\lambda)}{2}$
$(\kappa,\ \mu)$	$\kappa-\frac{2\mu}{3}$	
(μ,ν)	$\frac{2\mu\nu}{1-2\nu}$	
(λ,ν)		$\frac{\lambda(1-2\nu)}{2\nu}$
$(E,\ \nu)$	$\frac{E}{(1+\nu)(1-2\nu)}$	$\frac{E}{2(1+\nu)}$
$(\kappa,\ \nu)$	$\frac{3\kappa\nu}{1+\nu}$	$\frac{2\kappa(1-2\nu)}{2(1+\nu)}$
$(\kappa,\ E)$	$\frac{3\kappa(3\kappa-E)}{9\kappa-E}$	$\frac{3\kappa E}{9\kappa-E}$

Table 3.3: Conversion table for material parameters.

solved if we transform all the information to a single basis; we choose the fixed basis. The orientation of the unit cell relative to the fixed Cartesian basis $\boldsymbol{e}_i$ can be represented by the orthogonal matrix $\boldsymbol{R}$: that is,

$$\hat{\boldsymbol{e}}_i = \boldsymbol{R}\boldsymbol{e}_i,$$

The rotation matrix $\boldsymbol{R}$ can be broken down into three transformations about the z-, y- and z-axis $\boldsymbol{\Theta}_1$, $\boldsymbol{\Theta}_2$ and $\boldsymbol{\Theta}_3$ respectively. These rotations are given by

$$\boldsymbol{\Theta}_1 = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \boldsymbol{\Theta}_2 = \begin{pmatrix} \cos\theta_2 & 0 & \sin\theta_2 \\ 0 & 1 & 0 \\ -\sin\theta_2 & 0 & \cos\theta_2 \end{pmatrix} \boldsymbol{\Theta}_3 = \begin{pmatrix} \cos\theta_3 & -\sin\theta_3 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

where θ_1, θ_2 and θ_3 are the rotation angles about the z-, y- and z-axis. Thus, $\boldsymbol{R} = \boldsymbol{\Theta}_3\boldsymbol{\Theta}_2\boldsymbol{\Theta}_1$. The transformation from the unit cell basis to the fixed basis is given by [1]

$$\hat{\boldsymbol{e}}_\alpha = R\boldsymbol{e}_\alpha = R_{i\alpha}\boldsymbol{e}_i.$$

So a vector $\boldsymbol{v} = \hat{v}_\alpha\hat{\boldsymbol{e}}_\alpha$ given in the unit cell basis is transformed to the fixed basis $\boldsymbol{v} = v_i\boldsymbol{e}_i$ via

$$\boldsymbol{v} = \hat{v}_\alpha\hat{\boldsymbol{e}}_\alpha = \hat{v}_\alpha R_{i\alpha}\boldsymbol{e}_i = \underbrace{R_{i\alpha}\hat{v}_\alpha}_{v_i}\boldsymbol{e}_i.$$

[1] Here we use $\boldsymbol{T}\boldsymbol{e}_j = (T_{ik}\boldsymbol{e}_i \otimes \boldsymbol{e}_k)\boldsymbol{e}_j = T_{ik}\boldsymbol{e}_i\delta_{kj} = T_{ij}\boldsymbol{e}_i$.

This transformation must be applied to the vectors $\boldsymbol{s}^\alpha$ and $\boldsymbol{m}^\alpha$ defined in the unit cell basis. Similarly a fourth-order tensor $\mathcal{C}^e$, must be represented in the fixed basis as:

$$\begin{aligned}\mathcal{C}^e &= \hat{\mathcal{C}}^e_{\alpha\beta\gamma\delta}\ \hat{\boldsymbol{e}}_\alpha \otimes \hat{\boldsymbol{e}}_\beta \otimes \hat{\boldsymbol{e}}_\gamma \otimes \hat{\boldsymbol{e}}_\delta \\ &= \hat{\mathcal{C}}^e_{\alpha\beta\gamma\delta}\ R_{i\alpha}\boldsymbol{e}_i \otimes R_{j\beta}\ \boldsymbol{e}_j \otimes R_{k\gamma}\boldsymbol{e}_k \otimes R_{l\delta}\boldsymbol{e}_l \\ &= \underbrace{\hat{\mathcal{C}}^e_{\alpha\beta\gamma\delta}\ R_{i\alpha}R_{j\beta}R_{k\gamma}R_{l\delta}}_{\mathcal{C}^e_{ijkl}}\ \boldsymbol{e}_i \otimes \boldsymbol{e}_j \otimes \boldsymbol{e}_k \otimes \boldsymbol{e}_l.\end{aligned}$$

3.3 Constrained minimisation problems

In this section we consider methods for solving the constrained minimisation problem:

$$\begin{aligned}&\text{minimise } f(\boldsymbol{x}) \\ &\text{subject to } g_i(\boldsymbol{x}) \leq 0,\ \boldsymbol{x} \in \mathcal{S},\ \mathcal{S} \subset \mathbb{R}^n\end{aligned} \tag{3.23}$$

where $f, g_i : \mathcal{S} \to \mathbb{R}$ are continuous functions on the set $\mathcal{S}$. We focus on three optimisation methods for solving this problem. These are the penalty, Lagrange multiplier and augmented Lagrangian methods. Details on these methods can be found in [26].

3.3.1 Penalty method

One of the easiest methods to apply to constrained minimisation problems is the penalty method. We define a penalty function $P_i(\boldsymbol{x}) : \mathcal{S} \to \mathbb{R}$ with the following properties:

1. $P_i(\boldsymbol{x}) > 0$ for all $g_i(\boldsymbol{x}) > 0$,
2. $P_i(\boldsymbol{x}) = 0$ for all $g_i(\boldsymbol{x}) \leq 0$.

Now problem (3.23) can be written as the unconstrained optimisation problem

$$\text{minimise } \mathcal{L}(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_i \eta P_i(\boldsymbol{x}) \tag{3.24}$$

where η is a positive constant scalar penalty parameter.

A common example of a penalty function is

$$P_i(\boldsymbol{x}) = \max\left[0, g_i(\boldsymbol{x})\right]^2 .$$

It is important to note that a solution to (3.24) is not a solution to (3.23); rather it is an approximate solution to (3.23). One can show that as the penalty parameter increases, resulting in the constraint being more strongly imposed, the solution to (3.24) tends to the solution for (3.23); see [26, p. 367].

3.3.2 Lagrange multipliers

Lagrange multipliers for equality constraints

In order to understand how to solve problem (3.23) let us consider a simpler problem:

$$\begin{aligned} &\text{minimise } f(\boldsymbol{x}) \\ &\text{subject to } h_i(\boldsymbol{x}) = 0. \end{aligned} \tag{3.25}$$

This problem can be solved using Lagrange multipliers. A well known condition of a point $\boldsymbol{x}^*$, which is a solution to the problem, is the first-order necessary condition (see [26, p. 326]) at $\boldsymbol{x}^*$, the gradient of f is equal to the linear combination of the gradients of h_i and $h_i = 0$; that is,

$$\nabla f(\boldsymbol{x}^*) = -\sum_i \zeta_i \nabla h_i(\boldsymbol{x}^*), \tag{3.26a}$$

$$h_i(\boldsymbol{x}^*) = 0, \tag{3.26b}$$

where ζ_i are scalars called Lagrange multipliers. This leads to the Lagrange multiplier statement of problem (3.25)

$$\text{minimise } \mathcal{L}(\boldsymbol{x}, \zeta_i) = f + \sum_i \zeta_i h_i, \tag{3.27}$$

where the Lagrange multipliers are introduced as additional variables. One can see that a solution to (3.27) will also be a solution to (3.25), since at a solution to (3.27) we must have

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}} = 0 \rightarrow \nabla f = -\sum_i \zeta_i \nabla h_i$$
$$\frac{\partial \mathcal{L}}{\partial \zeta_i} = 0 \rightarrow h_i = 0$$

which are the requirements in (3.26). The problem statement (3.27) is said to be the dual problem of (3.25) in that a solution to the Lagrange multiplier problem is equivalent to a solution to the original problem.

Lagrange multipliers for inequality constraints

To solve problem (3.23) using the idea of Lagrange multipliers we need to add some further requirements on the multipliers. We start by defining active and inactive constraints. The i^{th} constraint is said to be active if the solution to the problem $\boldsymbol{x}^*$ satisfies $g_i(\boldsymbol{x}^*) = 0$. The i^{th} constraint is said to be inactive if $g_i(\boldsymbol{x}^*) < 0$.

If we knew the set of active constraints *a priori*, then we would ignore the inactive constraints and simply solve the problem only considering the active constraints:

minimise $f(\boldsymbol{x})$
subject to $g_i(\boldsymbol{x}) = 0$, for i in the active set.

Here we would simply apply Lagrange multipliers, as discussed in the previous section. Another way to achieve this is to associate Lagrange multiplier values $\zeta_i = 0$ with the inactive constraints. This renders the constraint inactive in that it has no effect on the minimisation. So we can say, if $g_i(x) < 0$ then $\zeta_i = 0$.

Now we turn our attention towards the active constraints. We lay out an argument for a single active constraint. Imagine at $\boldsymbol{x}^*$ the k^{th} constraint is the only active constraint, and has $\zeta_k < 0$. The first order-necessary condition implies that the gradient of f and the gradient of g_k are in the same direction. The gradient of g_k always points into the region where $g_k < 0$. Since the gradient of f also points in this direction, f could be further minimised in that direction. Hence, there is a point in the feasible region that further

minimises f. Thus the assumption that $\zeta_k < 0$ was incorrect. One can go further than this and show that this is true for multiple active constraints, see [26, pg 342]. Thus, if the k^{th} constraint is active, its associated Lagrange multiplier must be $\zeta_k > 0$.

This leads us to the KKT conditions. If we wish to solve problem (3.23) using a Lagrange multiplier approach, the following conditions apply:

$$\begin{aligned} \zeta_i &\geq 0 \\ g_i &\leq 0 \\ g_i\zeta_i &= 0. \end{aligned}$$

Thus problem (3.23) can be written as

$$\begin{aligned} &\text{minimise } \mathcal{L}(\boldsymbol{x}, \zeta_i) = f + \sum_i \zeta_i g_i \\ &\text{where } \zeta_i \geq 0, \qquad g_i \leq 0, \qquad \zeta_i g_i = 0. \end{aligned}$$

3.3.3 Augmented Lagrangian

Augmented Lagrangian for equality constraints

This method can be seen as a combination of a penalty method and a dual method (or Lagrange multiplier method). The augmented Lagrangian function for (3.25) is given by

$$\mathcal{L}_\eta(\boldsymbol{x}, \zeta_i) = f(\boldsymbol{x}) + \sum_i \zeta_i h_i(\boldsymbol{x}) + \tfrac{1}{2}\eta \sum_i |h_i(\boldsymbol{x})|^2 \tag{3.28}$$

for some some positive constant η. One can argue, see [26, pg 451], that this can be viewed as a special penalty function or as the basis for a dual problem.

From the view of a penalty function, we are simply using a standard quadratic penalty function on the problem

$$\begin{aligned} &\text{minimise } f(\boldsymbol{x}) + \sum_i \zeta_i h_i \\ &\text{subject to } h_i(\boldsymbol{x}) = 0, \end{aligned} \tag{3.29}$$

with fixed ζ_i. This problem has the same solution as (3.25) because adding a combination of the constraints will have no effect on the minima. If one were to select the values for ζ_i^* that correspond to the Lagrange multipliers at $\boldsymbol{x}^*$, then the gradient of $\mathcal{L}_\eta(\boldsymbol{x}, \zeta_i)$ would vanish at $\boldsymbol{x}^*$. This can be seen by differentiating $\mathcal{L}_\eta(\boldsymbol{x}, \zeta_i)$, and setting it to zero, to obtain

$$\nabla \mathcal{L}_\eta(\boldsymbol{x}, \zeta_i) = \nabla f(\boldsymbol{x}) + \sum_i \zeta_i \nabla h_i(\boldsymbol{x}) + \tfrac{1}{2}\eta \sum_i 2h_i(\boldsymbol{x}) \nabla h_i(\boldsymbol{x}) = 0$$

which would be satisfied by $\nabla f(\boldsymbol{x}) + \sum_i \zeta_i \nabla h_i(\boldsymbol{x}) = 0$ and $h_i(\boldsymbol{x}) = 0$. This is what is enforced by (3.29) if $\zeta_i = \zeta_i^*$.

An algorithm to find the minimum of (3.28) is given in [26, pg 452]. For a fixed value of ζ_i^k one finds the minimum of (3.28). Then ζ_i^k is updated using

$$\zeta_i^{k+1} = \zeta_i^k + \eta h_i(\boldsymbol{x}^k), \tag{3.30}$$

and (3.28) is minimised again with the updated ζ_i^{k+1}, and so on until convergence. One can also adjust the value of the penalty parameter η during the process. The way in which η is adjusted is usually predetermined and set to increase with more iterations.

It can be shown, in the context of the penalty method see Section 3.3.1, that the penalty function is an approximation of the Lagrange multiplier [26, pg 410]. In the context of this problem it is seen as the change in the Lagrange multiplier between iterations of problem (3.29). This serves as a justification for the form that the update formula (3.30) takes.

From the viewpoint of a dual problem, we are simply applying the method of Lagrange multipliers to

$$\text{minimise } f(\boldsymbol{x}) + \tfrac{1}{2}\eta \sum_i |h_i(\boldsymbol{x})|^2,$$

$$\text{subject to } h_i(\boldsymbol{x}) = 0.$$

Once again this problem has the same solution as (3.25) because the addition of the term $\frac{1}{2}\eta \sum_i |h_i(\boldsymbol{x})|^2$ has no effect on the minima, nor the Lagrange multipliers.

Augmented Lagrangian for inequality constraints

We now apply this approach to (3.23). We do this by converting the inequality constraints into equality constraints by introducing a slack variable z_i

$$\begin{aligned}&\text{minimise } f(\boldsymbol{x}),\\ &\text{subject to } g_i(\boldsymbol{x}) + z_i^2 = 0.\end{aligned}$$

Since the problem has been converted into one with only equality constraints, we can apply the same approach as in the previous section. We write the augmented Lagrangian function as before; that is,

$$\mathcal{L}_\eta(\boldsymbol{x}, \zeta_i, z_i) = f(\boldsymbol{x}) + \sum_i \underbrace{\left[\zeta_i(g_i(\boldsymbol{x}) + z_i^2) + \tfrac{1}{2}\eta \left|(g_i(\boldsymbol{x}) + z_i^2)\right|^2\right]}_{P_i}. \tag{3.31}$$

One can eliminate the dependence of $\mathcal{L}_\eta(\boldsymbol{x}, \zeta_i, z_i)$ on z_i. The only dependence of (3.31) on z_i is in the terms P_i. The derivative of P_i with respect to z_i is given by

$$\frac{\partial P_i}{\partial z_i} = 2\zeta_i z_i + 2\eta(g_i(\boldsymbol{x}) + z_i^2)z_i.$$

We note that $z_i^2 \geq 0$. If one is not on the i$^{\text{th}}$ constraint then one must have $z_i^2 > 0$, and the derivative of P_i must vanish. For this case it implies $z_i^2 = -\zeta_i/\eta - g_i(\boldsymbol{x})$. Thus one can write z_i^2 as

$$z_i^2 = \begin{cases} -\zeta_i/\eta - g_i(\boldsymbol{x}) & \text{if } -\zeta_i/\eta - g_i(\boldsymbol{x}) \geq 0,\\ 0 & \text{otherwise,}\end{cases}$$

or equivalently as

$$z_i^2 = \max\left[0, \ -\zeta_i/\eta - g_i(\boldsymbol{x})\right].$$

We now substitute this back into P_i. If $z_i^2 = 0$, the expression for P_i can be written as

$$P_i = \tfrac{1}{2\eta}\left[2\zeta_i\eta g_i(\boldsymbol{x}) + \eta^2 g^2(\boldsymbol{x})\right] = \tfrac{1}{2\eta}\left[(\zeta_i + \eta g_i(\boldsymbol{x}))^2 - \zeta_i^2\right].$$

If $z_i^2 = -\zeta_i/\eta - g_i(\boldsymbol{x})$, the expression for P_i can be written as

$$P_i = -\tfrac{1}{2\eta}\zeta_i^2.$$

These can be combined into a single statement:

$$P_i = \tfrac{1}{2\eta}\left\{\max\left[0,\ \zeta_i + \eta g_i(\boldsymbol{x})\right]^2 - \zeta_i^2\right\}.$$

Thus, the augmented Lagrangian function can be written as

$$\mathcal{L}_\eta(\boldsymbol{x}, \zeta_i) = f(\boldsymbol{x}) + \sum_i \tfrac{1}{2\eta}\left\{\left(\max\left[\zeta_i + \eta g_i(\boldsymbol{x})\right]\right)^2 - \zeta_i^2\right\}.$$

The minimiser of this Lagrangian is equivalent to a solution to (3.23).

3.4 Alternative representations of the flow rule for crystal plasticity

In Section 3.2.2 the flow rule was derived from the maximum plastic dissipation inequality. Here an alternative derivation of the flow rule is presented, based on the nonlinear programming methods discussed in Section 3.3. We present the results of the augmented Lagrangian, Lagrange multiplier and penalty methods applied to the principle of maximum plastic dissipation (3.21).

3.4.1 Lagrange multiplier method

The theory of Lagrange multipliers applied to the statement of maximum plastic dissipation (3.21) leads to the following problem:

Minimise: $\mathcal{L}_m(\boldsymbol{\sigma}, \chi^\alpha, \zeta^\alpha) = -\left(\boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}}^p + \sum_\alpha \dot{\lambda}^\alpha \chi^\alpha\right) + \sum_\alpha \zeta^\alpha \phi^\alpha$

subject to the KKT conditions

$\zeta^\alpha \geq 0, \quad \phi^\alpha \leq 0, \quad \phi^\alpha \zeta^\alpha = 0.$

Here we have introduced Lagrange multipliers ζ^α. Requiring stationarity of the Lagrangian with respect to the stress and hardening yields

$$\frac{\partial \mathcal{L}_m}{\partial \boldsymbol{\sigma}} = \mathbf{0} \Rightarrow \dot{\boldsymbol{\varepsilon}}^p = \sum_\alpha \zeta^\alpha \boldsymbol{P}^\alpha,$$
$$\frac{\partial \mathcal{L}_m}{\partial \chi^\alpha} = 0 \Rightarrow \dot{\lambda}^\alpha = \zeta^\alpha.$$

Thus, this treatment of the maximum plastic dissipation (3.21) leads to the flow rule

$$\dot{\boldsymbol{\varepsilon}}^p = \sum_\alpha \dot{\lambda}^\alpha \boldsymbol{P}^\alpha, \tag{3.32a}$$

$$\text{with conditions } \dot{\lambda}^\alpha \geq 0, \quad \phi^\alpha \leq 0, \quad \phi^\alpha \dot{\lambda}^\alpha = 0. \tag{3.32b}$$

As we can see by comparing the conditions (3.32) with (3.19)–(3.20), this treatment of the problem has a strict fulfillment of the complementarity conditions.

3.4.2 Penalty method

By directly applying the theory of the penalty method to the statement of maximum plastic dissipation (3.21) the following problem is obtained:

$$\text{Minimise } \mathcal{L}_p(\boldsymbol{\sigma}, \chi^\alpha) = -\left(\boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}}^p + \sum_\alpha \dot{\lambda}^\alpha \chi^\alpha\right) + \eta \sum_\alpha P(\phi^\alpha).$$

Here we have introduced the penalty parameter η, and an arbitrary penalty function $P(\phi^\alpha)$ that meets the requirements (see Section 3.3.1). Requiring stationarity of the Lagrangian with respect to the stress and hardening parameters yields

$$\frac{\partial \mathcal{L}_p}{\partial \boldsymbol{\sigma}} = \mathbf{0} \Rightarrow \dot{\boldsymbol{\varepsilon}}^p = \eta \sum_\alpha \frac{\partial P}{\partial \phi^\alpha} \frac{\partial \phi^\alpha}{\partial \boldsymbol{\sigma}} = \eta \sum_\alpha \frac{\partial P}{\partial \phi^\alpha} \boldsymbol{P}^\alpha,$$
$$\frac{\partial \mathcal{L}_p}{\partial \chi^\alpha} = 0 \Rightarrow \dot{\lambda}^\alpha = \eta \frac{\partial P}{\partial \phi^\alpha} \frac{\partial \phi^\alpha}{\partial \chi^\alpha} = \eta \frac{\partial P}{\partial \phi^\alpha}.$$

This result is substantially different to the Lagrange multiplier counterpart. Here there is no strict fulfillment of the yield criterion nor the complementarity conditions (3.20). A slip

system is considered active if

$$\frac{\partial P}{\partial \phi^\alpha} > 0.$$

Essentially what this method allows is for the yield criterion to be violated, which activates plastic flow and encourages the yield function to relax towards the yield surface. Due to the nature of penalty functions, the stronger the violation the stronger the encouragement. A choice of penalty function can be chosen such that a viscoplastic formulation is obtained [37]. If one chose

$$P(\phi^\alpha) = \frac{\tau_0^\alpha - \chi^\alpha}{p+1}\left(\frac{\max[0,\phi^\alpha] + \tau_0^\alpha - \chi^\alpha}{\tau_0^\alpha - \chi^\alpha}\right)^{p+1} - \max[0,\phi^\alpha] - \frac{\tau_0^\alpha - \chi^\alpha}{p+1}, \tag{3.33}$$

where p is a positive scalar parameter the *strain-rate-sensitivity* , it leads to

$$\frac{\partial P}{\partial \phi^\alpha} = \left(\frac{\max[0,\phi^\alpha]}{\tau_0^\alpha - \chi^\alpha} + 1\right)^p - 1.$$

This is a constitutive relationship of a well known viscoplastic formulation; see references within [29]. For this reason this treatment of the problem is considered a rate-dependent formulation. The general form of these flow rules are:

$$\dot{\boldsymbol{\varepsilon}}^p = \sum_\alpha \dot{\lambda}^\alpha \boldsymbol{P}^\alpha, \tag{3.34a}$$

$$\dot{\lambda}^\alpha = \eta \frac{\partial P}{\partial \phi^\alpha}. \tag{3.34b}$$

As stated in Section 3.3.1, the larger the choice of η the more strictly the constraint is imposed. Thus, as $\eta \to \infty$ the penalty method tends to exhibit behavior that is rate-independent.

3.4.3 Augmented Lagrangian method

By directly applying the theory of augmented Lagrangian to the statement of maximum plastic dissipation (3.21) the following problem is obtained:

$$\text{Minimise: } \mathcal{L}_r(\boldsymbol{\sigma}, \chi^\alpha, \zeta^\alpha) = -\left(\boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}}^p + \sum_\alpha \dot{\lambda}^\alpha \chi^\alpha\right) + \tfrac{1}{2\eta} \sum_\alpha \left\{ \max\left[0, \zeta^\alpha + \eta\phi^\alpha\right]^2 - (\zeta^\alpha)^2 \right\}.$$

Here we have introduced Lagrange multipliers ζ^α, and penalty parameter η. Requiring stationarity of the Lagrangian with respect to the stress, Lagrange multipliers and hardening parameter yields

$$\begin{aligned}
&\frac{\partial \mathcal{L}_r}{\partial \boldsymbol{\sigma}} = \mathbf{0} \Rightarrow \dot{\boldsymbol{\varepsilon}}^p = \sum_\alpha \max\left[0, \zeta^\alpha + \eta\phi^\alpha\right] \boldsymbol{P}^\alpha, \\
&\frac{\partial \mathcal{L}_r}{\partial \zeta^\alpha} = 0 \Rightarrow \zeta^\alpha = \max\left[0, \zeta^\alpha + \eta\phi^\alpha\right], \\
&\frac{\partial \mathcal{L}_r}{\partial \chi^\alpha} = 0 \Rightarrow \dot{\lambda}^\alpha = \max\left[0, \zeta^\alpha + \eta\phi^\alpha\right].
\end{aligned}$$

By comparing the above equations it is seen that $\zeta^\alpha = \dot{\lambda}^\alpha$. Thus this treatment of maximum plastic dissipation leads to the flow rule

$$\dot{\boldsymbol{\varepsilon}}^p = \sum_\alpha \dot{\lambda}^\alpha \boldsymbol{P}^\alpha, \tag{3.35a}$$

$$\dot{\lambda}^\alpha = \max\left[0, \dot{\lambda}^\alpha + \eta\phi^\alpha\right]. \tag{3.35b}$$

A slip system is considered active if

$$\max[0, \dot{\lambda}^\alpha + \eta\phi^\alpha] > 0. \tag{3.36}$$

Considering equation (3.35b) we note that if $\dot{\lambda}^\alpha > 0$, we must have $\phi^\alpha = 0$ for the equation to be satisfied. The equation also implies that when $\phi^\alpha < 0$ we have $\dot{\lambda}^\alpha = 0$. This means that this approach satisfies the complementarity conditions (3.20) exactly. For this reason this treatment of the problem is considered a rate-independent formulation.

Part II

Numerical methods and the development of algorithms

Chapter 4

Numerical methods

In this chapter we focus on numerical methods that are used throughout the remainder of this presentation. These methods are often used as tools, but it is necessary to be familiar with their details. We will discus two decomposition, the LU factorisation and the singular value decomposition (SVD), and how they are applied to solve systems of equations. The LU decomposition (with pivoting) is extensively used on systems of equations that are well-conditioned. However, the SVD is extensively used on systems of equations that are ill-conditioned. In addition to these decomposition, we discuss a Newton–Raphson method that includes line search. This is know as a globally convergent Newton–Raphson method. Finaly, we briefly discus fixed point mapping. This topic is briefly discussed so as to familiarise the reader with the topic, and is not seen as a complete dicussion.

4.1 LU factorisation

Let $A \in \mathbb{R}^{m \times m}$ be a square matrix. A series of Gaussian elimination operations $L_1, L_2, ...L_{m-1}$ can be applied to A such that an upper triangular matrix U is formed:

$$\underbrace{L_{m-1}...L_2L_1}_{L^{-1}} A = U.$$

This can be expressed as

$$A = LU,$$

where L is unit lower triangular and U is upper triangular. This is known as the LU decomposition. Variations of this decomposition exist, such as the LDU decomposition where D is diagonal, L is unit lower triangular and U is unit upper triangular.

L can be shown to be lower triangular as follows. Consider the k$^{\text{th}}$ Gaussian elimination operation L_k. Let a_k be the k$^{\text{th}}$ column of A before L_k is applied. L_k must be chosen such that

$$a_k = \begin{bmatrix} a_{1k} \\ \vdots \\ a_{kk} \\ a_{k+1,k} \\ \vdots \\ a_{mk} \end{bmatrix} \xrightarrow{L_k} L_k a_k = \begin{bmatrix} a_{1k} \\ \vdots \\ a_{kk} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

This is done by subtracting l_{jk} times row k from row j, where l_{jk} is given by

$$l_{jk} = \frac{a_{jk}}{a_{kk}} \qquad k < j \leq m.$$

The matrix L_k is then given by

$$L_k = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & l_{k+1,k} & 1 & \\ & & \vdots & & \ddots \\ & & l_{m,k} & & 1 \end{bmatrix}$$

from which we define

$$l_k = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ l_{k+1,k} \\ \vdots \\ l_{m,k} \end{bmatrix}.$$

L_k can be expressed as $L_k = I - l_k e_k^T$ where e_k is a vector with 1 in position k and zero elsewhere. Now consider $(I - l_k e_k^T) \cdot (I + l_k e_k^T) = I - l_k \otimes e_k \cdot l_k \otimes e_k = I - (l_k \cdot e_k) l_k \otimes e_k = I$, since $(l_k \cdot e_k) = 0$. This implies that $L_k^{-1} = I + l_k \otimes e_k$. Next, consider

$$\begin{aligned} L_k^{-1} L_{k+1}^{-1} &= (I + l_k \otimes e_k) \cdot (I + l_{k+1} \otimes e_{k+1}) \\ &= I + l_k \otimes e_k + l_{k+1} \otimes e_{k+1} + (l_k \otimes e_k) \cdot (l_{k+1} \otimes e_{k+1}) \\ &= I + l_k \otimes e_k + l_{k+1} \otimes e_{k+1} + (e_k \cdot l_{k+1})(l_k \otimes e_{k+1}) \\ &= I + l_k \otimes e_k + l_{k+1} \otimes e_{k+1} \end{aligned}$$

since $(e_k \cdot l_{k+1}) = 0$. Thus

$$L = L_1^{-1} L_2^{-1} ... L_{m-1}^{-1} = I + l_1 \otimes e_1 + \cdots + l_{m-1} \otimes e_{m-1} = \begin{bmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ l_{m1} & l_{m2} & \dots & l_{m,m-1} & 1 \end{bmatrix}.$$

Therefore, L is in fact unit lower triangular.

4.1.1 Remarks on using LU decomposition to solve the system $Ax = b$.

LU decomposition is often used to solve the system of equations $Ax = b$. This is done by factorising $A = LU$, and solving the systems $Ly = b$ and $Ux = y$. As illustrated in [43, pg 153], this form of the LU decomposition does not solve $Ax = b$ stably. In general, Gaussian elimination plus pivoting is necessary for a LU decomposition that is stable in solving the

problem $Ax = b$, see [43, pg 155]. The expansion of this to include pivoting is implemented here.

4.2 Singular value decomposition

Singular value decomposition can be illustrated by the following geometric fact [43]: *The image of the unit sphere under any $m \times n$ matrix is a hyperellipse.* Let the unit sphere of dimension n be represented by the orthonormal set $\{v_j\}$. Then under the action of $\boldsymbol{A}$ (an $m \times n$ matrix) $\{v_j\}$ is mapped to $\{\sigma_i u_i\}$, where $\{u_i\}$ are the directions of the principal semiaxes of the hyperellipse, and σ_i are the lengths of the principal semiaxis of the hyperellipse. Since the principal semiaxes are orthogonal, $\{u_i\}$ form an othonormal set of vectors. The mapping of one of the vectors in $\{v_j\}$ can be represented by $A_{ik}v_k = \sigma_i u_i$. The mapping of the entire set $\{v_j\}$ can be represented by $A_{ik}v_{kj} = \sigma_j u_{ik}\delta_{kj}$. This leads to the matrix representation

$$\boldsymbol{AV} = \boldsymbol{U\Sigma}.$$

Since $\boldsymbol{V}$ consists of orthonormal columns it is orthogonal and $\boldsymbol{V}^{-1} = \boldsymbol{V}^T$. So,

$$\boldsymbol{A} = \boldsymbol{U\Sigma V}^T.$$

This is known as the reduced singular value decomposition, and is sufficient for our purposes. The inverse of $\boldsymbol{A}$ can easily be determined. Since $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary, their inverses are equal to their transposes. $\boldsymbol{\Sigma}$ is diagonal ($\boldsymbol{\Sigma} = \sigma_j\delta_{kj} = \text{diag}(\sigma_i)$) so its inverse is a diagonal matrix with diagonal components equal to the reciprocals of σ_i:

$$\boldsymbol{A}^{-1} = \left(\boldsymbol{U\Sigma V}^T\right)^{-1} = \boldsymbol{V}[\text{diag}(1/\sigma_j)]\boldsymbol{U}^T.$$

Consider the problem

$$\boldsymbol{Ax} = \boldsymbol{b}. \tag{4.1}$$

A problem arises in finding solutions $\boldsymbol{x}$ to this equation when $\boldsymbol{A}$ is singular. If $\boldsymbol{A}$ is singular then there are solutions to a $\boldsymbol{Ax} = \boldsymbol{0}$ other than $\boldsymbol{x} = \boldsymbol{0}$. The set of these solutions is called

the *null space* of $\boldsymbol{A}$. The dimension of the null space is called the *nullity* of $\boldsymbol{A}$. The subspace of $\boldsymbol{b}$ that can reached by $\boldsymbol{A}$ is called the *range* of $\boldsymbol{A}$. The dimension of the range is called the *rank* of $\boldsymbol{A}$. In SVD we explicitly construct an orthonormal basis of the null space and the range of $\boldsymbol{A}$. The columns of $\boldsymbol{U}$ that have associated non-zero singular values span the range of $\boldsymbol{A}$. The columns of $\boldsymbol{V}$ that have associated singular values that are zero span the null space of $\boldsymbol{A}$. Using this last statement we can solve (4.1) when $\boldsymbol{A}$ is singular and $\boldsymbol{b}$ zero, since the solution to this problem is given immediately by SVD - the columns of $\boldsymbol{V}$ with associated zero singular values.

Now we discuss solving equation (4.1) when $\boldsymbol{A}$ is singular and $\boldsymbol{b}$ is in the range of $\boldsymbol{A}$. This problem has more than one solution $\boldsymbol{x}$, being a linear combination of $\boldsymbol{x}$ and any other vector in the null space. If we want to produce a single vector that is a solution, we might choose the one that has the smallest length $|\boldsymbol{x}|$. We achieve this by replacing $1/\sigma_i$ by 0 if $\sigma_i = 0$. Hence

$$\boldsymbol{x} = \boldsymbol{V}\hat{\boldsymbol{\Sigma}}^{-1}\boldsymbol{U}^T\boldsymbol{b}, \tag{4.2}$$

where $\hat{\boldsymbol{\Sigma}}^{-1}$ is the modified inverse of $\boldsymbol{\Sigma}$.

When $\boldsymbol{b}$ is not in the range of $\boldsymbol{A}$ then equation (4.1) has no solution. But by using equation (4.2) we can still find a "solution". This "solution" will not satisfy equation (4.1) exactly, instead it will do the best possible job in a least squares sense. In other words it is the solution to

$$\boldsymbol{x} \quad \text{which minimises} \quad |\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}|.$$

Above we considered a solution scheme when $\boldsymbol{A}$ is singular. This scheme also works for $\boldsymbol{A}$ when it is ill-conditioned.

4.3 Newton–Raphson method

The Newton–Raphson method is a root-finding technique, see [34, pg 379]. The problem can be stated as follows:

Find $\boldsymbol{x} \in \mathbb{R}^n$ such that $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}$, where $\boldsymbol{F} : \mathbb{R}^n \to \mathbb{R}^n$.

In the neighbourhood of a solution to this problem, $\boldsymbol{F}$ can be expanded in the truncated Taylor series

$$\boldsymbol{F}(\boldsymbol{x}+\delta\boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{x}) + \frac{\partial \boldsymbol{F}(\boldsymbol{x})}{\partial \boldsymbol{x}}\delta\boldsymbol{x} + \mathcal{O}(\delta\boldsymbol{x}^2).$$

The term $\boldsymbol{J} := \frac{\partial \boldsymbol{F}(\boldsymbol{x})}{\partial \boldsymbol{x}}$ is called the *Jacobian*. By setting $\boldsymbol{F}(\boldsymbol{x}+\delta\boldsymbol{x}) = \boldsymbol{0}$ and neglecting terms of $\mathcal{O}(\delta\boldsymbol{x}^2)$, an expression for $\delta\boldsymbol{x}$ is obtained in the direction of the root

$$\boldsymbol{J}(\boldsymbol{x})\delta\boldsymbol{x} = -\boldsymbol{F}(\boldsymbol{x}).$$

When solving this system of equations the nature of $\boldsymbol{J}$ should be considered. If $\boldsymbol{J}$ is well behaved then a solution can be found using LU decomposition[1]. However, if $\boldsymbol{J}$ is ill-conditioned then a solution can be found using singular value decomposition[2]. Once the system of equations is solved the correction $\delta\boldsymbol{x}$ is added to the previous approximation to obtain

$$\boldsymbol{x}_{\text{new}} = \boldsymbol{x}_{\text{old}} + \delta\boldsymbol{x}$$

and the process is iterated until convergence. In practice two checks for convergence are used: one criterion checks that a root is sufficiently close, $||\boldsymbol{F}||_\infty < \text{tol}$, another checks that the change in the position of the root is significant, $||\delta\boldsymbol{x}||_\infty < \text{tol}$.

This Newton–Raphson method works very well provided $\boldsymbol{F}$ and $\boldsymbol{J}$ are well defined and the starting point is sufficiently close to the root. However, in many cases we require a root-finding technique that is more robust: one that is globally convergent, and that can handle irregularities in $\boldsymbol{F}$.

4.3.1 Globally convergent Newton–Raphson method: Newton–Raphson combined with line search

In this Section we discuss a root-finding technique that combines the rapid local convergence of a Newton–Raphson method with a strategy that ensures some progression is made in the direction of the root. The argument presented here can be found in [34, pg 387]. A strategy

[1] see Section 4.1
[2] see Section 4.2

to achieve a globally convergent Newton–Raphson is to add an additional constraint. This constraint says that the Newton step $\delta\boldsymbol{x}$ must *reduce* $f = 1/2|\boldsymbol{F}|^2 = 1/2\boldsymbol{F}\cdot\boldsymbol{F}$. What must be done if the Newton step does not reduce f? We note that $\delta\boldsymbol{x}$ is in the direction that reduces f in the vicinity of the current position; that is

$$\nabla f \cdot \delta\boldsymbol{x} = (\boldsymbol{F}\cdot\boldsymbol{J})\cdot\left(-\boldsymbol{J}^{-1}\cdot\boldsymbol{F}\right) = -\boldsymbol{F}\cdot\boldsymbol{F} \leq 0. \tag{4.3}$$

Thus, there exists a $\lambda \in (0,1]$ such that the update $\boldsymbol{x}_{\text{new}} = \boldsymbol{x}_{\text{old}} + \lambda\delta\boldsymbol{x}$ reduces f.

The next question is how do we find a λ that reduces f? This can be done by the using the line search technique. The idea is to approximate the function $g(\lambda) = f(\boldsymbol{x}_{old} + \lambda\delta\boldsymbol{x})$ by successive polynomials and find the minimum of these until a criterion is met. In practice the following criterion is normally used:

$$f(\boldsymbol{x}_{old} + \lambda\delta\boldsymbol{x}) \leq f(\boldsymbol{x}_{old}) + \alpha\nabla f(\boldsymbol{x}_{old})\cdot[(\boldsymbol{x}_{old} + \lambda\delta\boldsymbol{x}) - \boldsymbol{x}_{old}], \tag{4.4}$$

where $\alpha \in (0,1)$ is a parameter. Within the context of a Newton–Raphson with line search if $\boldsymbol{F}$ is of order unity then a value of $\alpha = 10^{-4}$ is appropriate. This criterion is better than the criterion $f(\boldsymbol{x}_{old} + \lambda\delta\boldsymbol{x}) < f(\boldsymbol{x}_{old})$, as this criterion can fail to converge to the minimum of f.

The first polynomial approximation of $g(\lambda)$ is the quadratic function

$$g(\lambda) = \left[g(1) - g(0) - g'(0)\right]\lambda^2 + g'(0)\lambda + g(0), \tag{4.5}$$

where $g'(\lambda) = \nabla f\cdot\delta\boldsymbol{x}$. The minimum of this polynomial is a good appropriation for the λ that satisfies (4.4), and is given by

$$\lambda = \frac{-g'(0)}{2\left[g(1) - g(0) - g'(0)\right]}\,.$$

Now we can change the second and successive polynomial approximations to a cubic approximation. In this approximation we use the previous values $g(\lambda_1)$ and $g(\lambda_2)$, where λ_1 and λ_2 are the previous two approximations of the minimum (λ_1 being the most recent approximation). That is,

$$g(\lambda) = a\lambda^3 + b\lambda^2 + g'(0) + g(0),$$

where a and b are given by

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\lambda_1 - \lambda_2} \begin{bmatrix} 1/\lambda_1^2 & -1/\lambda_2^2 \\ -\lambda_2/\lambda_1^2 & \lambda_1/\lambda_2^2 \end{bmatrix} \cdot \begin{bmatrix} g(\lambda_1) - g'(0)\lambda_1 - g(0) \\ g(\lambda_2) - g'(0)\lambda_2 - g(0) \end{bmatrix}. \tag{4.6}$$

The minimum of this cubic equation is given by

$$\lambda = \frac{-b + \sqrt{b^2 - 3ag'(0)}}{3a}.$$

In practice a constraint that $0.5\lambda_1 \leq \lambda \leq 0.1\lambda_1$ is placed on λ. This ensures that the system remains stable.

This process is continued until a λ that satisfies the criterion (4.4) is found, from which it is concluded that the next Newton step is taken to the point $\boldsymbol{x}_{new} = \boldsymbol{x}_{old} + \lambda\delta\boldsymbol{x}$.

4.4 Fixed point mappings

Let the set X be a normed space. A fixed point of a mapping $T : X \to X$ is a point $x \in X$, such that $Tx = x$. Of course not every mapping of this kind possesses a fixed point. There are certain criteria that a mapping must meet, one such theorm that state the criteria is the *Banach fixed point theorem*. This theorem states that if T is a *contraction mapping* on X, then T has precisely one fixed point, see [23].

The definition of contraction map is as follows: the mapping $T : X \to X$ is called a contraction mapping if there exists a positive scalar $\alpha < 1$ such that for any $x, y \in X$

$$||Tx - Ty|| \leq \alpha||x - y||.$$

A geometrical interpretation of this is that the image of the points x and y are closer together than x and y.

The Banach fixed point theorem also provides a means of finding the fixed point of a contraction map. This is done by starting at an arbitrary point $x_0 \in X$. Then by the iterative procedure

$$x \xleftarrow{T} x,$$

the fixed point of T can be found.

Chapter 5

Finite element method

Here we focus on applying the finite element method to the equations developed in Chapter 3 that govern the response of a continuum. The finite element method is a well established numerical method, and it is widely used in engineering applications. The topic is very mature and many different approaches to understanding this topic can be found in the literature, see for example Hughes [20], Reddy [35], Belytschko [10]. The application of the finite element method to elastoplastic materials can be found in Simo and Hughes [39].

5.1 The strong formulation

Here we consider the result of balance of linear momentum in the absence internal forces: the equilibrium condition (2.10). This equation along with the boundary conditions discussed in Section 2.4 make up what is known as the strong form of the problem: Given $\boldsymbol{b} : \Omega \to \mathbb{R}^3$, $\boldsymbol{g} : \Gamma_D \to \mathbb{R}^3$ and $\boldsymbol{h} : \Gamma_N \to \mathbb{R}^3$, find $\boldsymbol{u} : \Omega \to \mathbb{R}^3$ such that

$$\begin{aligned} \operatorname{div}(\boldsymbol{\sigma}) + \rho\boldsymbol{b} &= \boldsymbol{0} \text{ in } \Omega, \\ \boldsymbol{\sigma}\boldsymbol{n} &= \boldsymbol{h} \text{ on } \Gamma_N, \\ \boldsymbol{u} &= \boldsymbol{g} \text{ on } \Gamma_D, \end{aligned}$$

where $\boldsymbol{\sigma}(\boldsymbol{u}) = \mathcal{C}^e(\boldsymbol{\varepsilon}(\boldsymbol{u}) - \boldsymbol{\varepsilon}^p)$ together with the equations for plastic flow in Table 3.2. Here $\boldsymbol{\varepsilon}^p$ is not an explicit function, but is dependent on the dispacement and the history of the

plastic strain. This also applies to $\boldsymbol{\sigma}$.

5.2 The weak formulation

Let $\mathcal{S}$ denote the trial solution space, the set of candidate solutions. Let $\mathcal{V}$ denote the test space, a function space used to test candidate solutions. Each member $u_i \in \mathcal{S}$ must satisfy $u_i = g_i$ on Γ_D. We define that each $w_i \in \mathcal{V}$ must satisfy $w_i = 0$ on Γ_D. We use these test functions to construct the weak formulation. This is done by multiplying the equilibrium condition by $\boldsymbol{w}$ and integrating over the domain Ω:

$$\int_\Omega [\boldsymbol{w} \cdot \operatorname{div}(\boldsymbol{\sigma}) + \rho \boldsymbol{w} \cdot \boldsymbol{b}]\, dx = 0.$$

Using the divergence theorem on the first term we obtain

$$\int_\Gamma \boldsymbol{w} \cdot \boldsymbol{\sigma} \boldsymbol{n}\, ds - \int_\Omega \nabla \boldsymbol{w} : \boldsymbol{\sigma}\, dx + \int_\Omega \rho \boldsymbol{w} \cdot \boldsymbol{b}\, dx = 0.$$

Naturally the integral over the boundary can be separated into two boundary integrals, one over the Neumann boundary and one over the Dirichlet boundary:

$$\int_{\Gamma_N} \boldsymbol{w} \cdot \boldsymbol{\sigma} \boldsymbol{n}\, ds + \int_{\Gamma_D} \boldsymbol{w} \cdot \boldsymbol{\sigma} \boldsymbol{n}\, ds - \int_\Omega \nabla \boldsymbol{w} : \boldsymbol{\sigma}\, d\Omega + \int_\Omega \rho \boldsymbol{w} \cdot \boldsymbol{b}\, dx = 0.$$

We have defined the test function to be zero on the Dirichlet boundary (Γ_D), so the second term is eliminated. We also know that on the Neumann boundary Γ_N we have $\boldsymbol{\sigma} \boldsymbol{n} = \boldsymbol{h}$; thus

$$\int_\Omega \nabla \boldsymbol{w} : \boldsymbol{\sigma}\, dx = \int_{\Gamma_N} \boldsymbol{w} \cdot \boldsymbol{h}\, ds + \int_\Omega \rho \boldsymbol{w} \cdot \boldsymbol{b}\, dx. \tag{5.1}$$

Since $\boldsymbol{\sigma}$ is symmetric, we can write the first term of (5.1) as [1]

$$\nabla \boldsymbol{w} : \boldsymbol{\sigma} = \operatorname{sym}(\nabla \boldsymbol{w}) : \boldsymbol{\sigma}.$$

[1] Let A_{ij} be symmetric and B_{ij} be decomposed into a symmetric and anti-symmetric parts $B_{(ij)}$ and $B_{]ij[} = \omega_k \epsilon_{ijk}$, where ϵ_{ijk} is the perturbation tensor. Then $A_{ij}B_{ij} = A_{ij}(B_{(ij)} + B_{]ij[}) = A_{ij}B_{(ij)} + A_{ij}\omega_k\epsilon_{ijk} = A_{ij}B_{(ij)}$.

Thus the weak formulation can be stated as: Given $\boldsymbol{b} : \Omega \to \mathbb{R}^3$, $\boldsymbol{g} : \Gamma_D \to \mathbb{R}^3$ and $\boldsymbol{h} : \Gamma_N \to \mathbb{R}^3$, find $\boldsymbol{u} \in \mathcal{S}$ such that for all $\boldsymbol{w} \in \mathcal{V}$

$$\int_\Omega \text{sym}(\nabla \boldsymbol{w}) : \boldsymbol{\sigma}\, dx = \int_{\Gamma_N} \boldsymbol{w} \cdot \boldsymbol{h}\, ds + \int_\Omega \rho \boldsymbol{w} \cdot \boldsymbol{b}\, dx \tag{5.2}$$

where $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{u})$.

We note at this point that limitations must be placed on the spaces $\mathcal{S}$ and $\mathcal{V}$. Equation (5.1) requires that the members and their gradients are square integrable on Ω; thus by definition of the Sobolev space H^1,

$$\begin{aligned}
\mathcal{S} &= \left\{ \boldsymbol{u} \,|\, \boldsymbol{u} = \boldsymbol{g} \text{ on } \Gamma_D,\, \boldsymbol{u} \in [L^2(\Omega)]^3,\, \nabla \boldsymbol{u} \in [L^2(\Omega)]^{3\times 3} \right\} \\
&= \left\{ \boldsymbol{u} \,|\, \boldsymbol{u} = \boldsymbol{g} \text{ on } \Gamma_D,\, \boldsymbol{u} \in [H^1(\Omega)]^3 \right\}, \\
\mathcal{V} &= \left\{ \boldsymbol{w} \,|\, \boldsymbol{w} = \boldsymbol{0} \text{ on } \Gamma_D,\, \boldsymbol{w} \in [L^2(\Omega)]^3,\, \nabla \boldsymbol{w} \in [L^2(\Omega)]^{3\times 3} \right\} \\
&= \left\{ \boldsymbol{w} \,|\, \boldsymbol{w} \in [H^1_D(\Omega)]^3 \right\}.
\end{aligned}$$

where $\boldsymbol{g}$ is a given smooth function. One can also infer from $\mathcal{V} = [H^1_D(\Omega)]^3$ and $\mathcal{S} = [H^1(\Omega)]^3$ that $\mathcal{V} \subset \mathcal{S}$, with the restriction that it is zero on the Dirichlet boundary.

It can be shown that if all functions are smooth enough, a solution to the weak form is also a solution to the strong form [20]. This implies that the strong form is equivalent to the weak form. For convenience we define the notation

$$\begin{aligned}
a(\boldsymbol{w}, \boldsymbol{u}) &= \int_\Omega \text{sym}(\nabla \boldsymbol{w}) : \boldsymbol{\sigma}(\boldsymbol{u})\, dx, \\
(\boldsymbol{w}, \boldsymbol{b}) &= \int_\Omega \rho \boldsymbol{w} \cdot \boldsymbol{b}\, dx, \\
(\boldsymbol{w}, \boldsymbol{h})_\Gamma &= \int_{\Gamma_N} \boldsymbol{w} \cdot \boldsymbol{h}\, ds.
\end{aligned} \tag{5.3}$$

The weak form in this notation is then

$$a(\boldsymbol{w}, \boldsymbol{u}) = (\boldsymbol{w}, \boldsymbol{b}) + (\boldsymbol{w}, \boldsymbol{h})_\Gamma.$$

5.3 The Galerkin approximation

Next we construct the Galerkin approximation from the weak formulation. Let $\mathcal{S}^h$ and $\mathcal{V}^h$ be finite-dimensional approximations of $\mathcal{S}$ and $\mathcal{V}$. In particular $\boldsymbol{w}^h \in \mathcal{V}^h$ satisfies $\boldsymbol{w}^h = \boldsymbol{0}$ on Γ_D. We assume that the members of $\mathcal{S}^h$ can be decomposed into

$$\boldsymbol{u}^h = \boldsymbol{v}^h + \boldsymbol{g}^h,$$

where $\boldsymbol{v}^h \in \mathcal{V}^h$ and $\boldsymbol{g}^h$ satisfies the boundary condition $\boldsymbol{u}^h = \boldsymbol{g}^h$ on Γ_D. This allows for the expansion of the term $a(\boldsymbol{w}^h, \boldsymbol{u}^h) = a(\boldsymbol{w}^h, \boldsymbol{v}^h + \boldsymbol{g}^h) = a(\boldsymbol{w}^h, \boldsymbol{v}^h) + a(\boldsymbol{w}^h, \boldsymbol{g}^h)$.

Then the Galerkin formulation of the problem is: Given $\boldsymbol{b} : \Omega \to \mathbb{R}^3$, $\boldsymbol{g}^h : \Gamma_D \to \mathbb{R}^3$ and $\boldsymbol{h} : \Gamma_N \to \mathbb{R}^3$, find $\boldsymbol{u}^h = \boldsymbol{v}^h + \boldsymbol{g}^h \in \mathcal{S}^h$ such that for all $\boldsymbol{w} \in \mathcal{V}^h$

$$a(\boldsymbol{w}^h, \boldsymbol{v}^h) = (\boldsymbol{w}^h, \boldsymbol{b}) + (\boldsymbol{w}^h, \boldsymbol{h})_\Gamma - a(\boldsymbol{w}^h, \boldsymbol{g}^h) \tag{5.4}$$

where $\mathcal{S}^h$ and $\mathcal{V}^h$ are finite-dimensional sets and $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{u}^h)$.

Looking at (5.4), we realise that due to the decomposition of $\boldsymbol{u}^h$, the "unknowns" of the problem have been reduced to being in the space $\mathcal{V}^h$. From this point on we shall only discuss the space $\mathcal{V}^h$.

If Ξ is a finite-dimensional space, then there exists a basis $\boldsymbol{\psi}_A(\boldsymbol{x})$ such that any member $\boldsymbol{\xi} \in \Xi$ can be represented as the linear combination

$$\boldsymbol{\xi} = \sum_{A=1}^{\dim(\Xi)} \sum_{j=1}^{n} \alpha_{Aj} \psi_A(\boldsymbol{x}) \boldsymbol{e}_j = \sum_{A=1}^{\dim(\Xi)} \boldsymbol{\alpha}_A \boldsymbol{\psi}_A(\boldsymbol{x}), \tag{5.5}$$

where $\boldsymbol{\alpha}_A$ are scalars.

5.4 The finite element approximation

Next we focus our attention on the choice of basis for the finite-dimensional space $\mathcal{V}^h$. Here we will choose a standard conforming finite element basis.

Essential to this idea is that of partitioning $\bar{\Omega}$. We partition the domain $\bar{\Omega}$ into sub-domains

$\bar{\Omega}^e$ called *elements*, such that

$$\bigcup_e \bar{\Omega}^e = \bar{\Omega} \qquad \text{and} \qquad \bigcap_e \Omega^e = \emptyset.$$

Next we introduce special points called *nodes* within the partitioned domain. These points are placed at least at the vertices of the elements. In addition they can, in general, be placed on element boundaries, and within an element. We number the nodes $1, \ldots,$ *n_nodes* and denote node A as $\boldsymbol{x}_A$. These nodes form the set υ. We denote the points that lie on Γ as υ_g, thus those that lie in Ω are in υ/υ_g.

Now we focus on the basis functions themselves. In a finite element context these are called *finite element functions* or *shape functions*. We choose the following properties for our basis functions:

1. $\mathcal{V}^h$ to be in $C^0(\Omega)$, this complies with the earlier conclusion that $\mathcal{V}^h \in H^1_D(\Omega)$. We choose $\psi_A(\boldsymbol{x})$ to be continuous, piecewise polynomials such that

$$\psi_A(\boldsymbol{x})|_{\Omega^e} := \psi_A^{(e)}, \quad \psi_A^{(e)} \in P^n(\Omega^e), \tag{5.6}$$

   where P^n is the space of all polynomial of order less than or equal to n.

2. Each ψ_A is associated with a node $\boldsymbol{x}_A$, such that

$$\psi_A(\boldsymbol{x})|_{\bar{\Omega}^e} = 0 \text{ if } \boldsymbol{x}_A \notin \bar{\Omega}^e. \tag{5.7}$$

3. That they possess a Kronecker delta property, i.e. the function is zero at all nodes other than the node it is associated with and one at the associated node;

$$\psi_A(\boldsymbol{x}_B) = \begin{cases} 0 & \text{if } A \neq B \\ 1 & \text{if } A = B. \end{cases} \tag{5.8}$$

Many of the assumptions we make above will lead to a system that has beneficial properties in the context of implementation. See Figure 5.1 for a visual representation of an example basis function.

We consider the application of the bilinear form to $\sum_{l=1}^{3} \sum_B a(\psi_A \boldsymbol{e}_k, \alpha_{Bl} \psi_B \boldsymbol{e}_l)$. We rep-

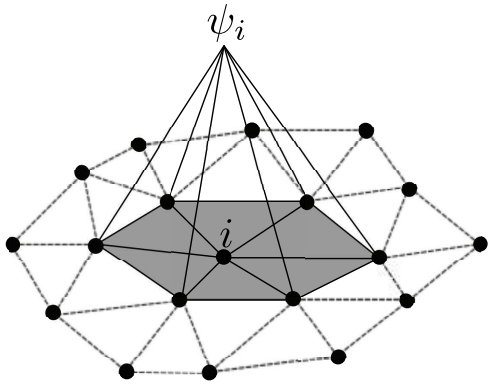


Figure 5.1: An example basis function associated with node i at position $\boldsymbol{x}_i$.

resent the spacial dimension of the problem by n_dim, in our case $n\_dim = 3$. Due to the summation properties of integration, an integral over $\bar{\Omega}$ can be broken up into integrals over $\bar{\Omega}^e$. Let $\mathcal{F}$ represent the integrand, then

$$\begin{aligned}\sum_{l=1}^{n\_dim}\sum_{B} a(\psi_A \boldsymbol{e}_k, \alpha_{Bl}\psi_B \boldsymbol{e}_l) &= \sum_{l=1}^{n\_dim}\sum_{B}\int_{\bar{\Omega}} \mathcal{F}(\psi_A \boldsymbol{e}_k, \alpha_{Bl}\psi_B \boldsymbol{e}_l)d\Omega \\ &= \sum_{e=1}^{N}\sum_{l=1}^{n\_dim}\sum_{B}\int_{\bar{\Omega}^e} \mathcal{F}(\psi_A \boldsymbol{e}_k, \alpha_{Bl}\psi_B \boldsymbol{e}_l)d\Omega \\ &= \sum_{e=1}^{N}\sum_{l=1}^{n\_dim}\sum_{B}\int_{\bar{\Omega}^e} \mathcal{F}(\psi_A^{(e)} \boldsymbol{e}_k, \alpha_{Bl}\psi_B^{(e)} \boldsymbol{e}_l)d\Omega,\end{aligned}$$

for which the notation

$$\begin{aligned}\sum_{e=1}^{N}\sum_{l=1}^{n\_dim}\sum_{B} a^{(e)}(\psi_A^{(e)} \boldsymbol{e}_k, \alpha_{Bl}\psi_B^{(e)} \boldsymbol{e}_l) &:= \sum_{e=1}^{N}\sum_{l=1}^{n\_dim}\sum_{B}\int_{\bar{\Omega}^e} \mathcal{F}(\psi_A^{(e)} \boldsymbol{e}_k, \alpha_{Bl}\psi_B^{(e)} \boldsymbol{e}_l)d\Omega \\ &= \sum_{l=1}^{n\_dim}\sum_{B} a(\psi_A \boldsymbol{e}_k, \alpha_{Bl}\psi_B \boldsymbol{e}_l)\end{aligned} \tag{5.9}$$

is defined.

We can also consider the application of the inner product of a basis function and a function

$\boldsymbol{h}$; that is $(\boldsymbol{\psi}_A, \boldsymbol{h}) = (\psi_A \boldsymbol{e}_k, \boldsymbol{h})$. Let $\mathcal{G}$ represent the integrand, then

$$\begin{aligned}(\psi_A \boldsymbol{e}_k, \boldsymbol{h}) &= \int_{\bar{\Omega}} \mathcal{G}(\psi_A \boldsymbol{e}_k) d\Omega \\ &= \sum_{e=1}^{N} \int_{\bar{\Omega}^e} \mathcal{G}(\psi_A \boldsymbol{e}_k) d\Omega \\ &= \sum_{e=1}^{N} \int_{\bar{\Omega}^e} \mathcal{G}(\psi_A^{(e)} \boldsymbol{e}_k) d\Omega,\end{aligned}$$

where

$$\boldsymbol{h}^{(e)}(\boldsymbol{x}) = \begin{cases} \boldsymbol{0} & \text{if } \boldsymbol{x} \notin \Omega^e \\ \boldsymbol{h} & \text{if } \boldsymbol{x} \in \Omega^e. \end{cases}$$

For this the notation

$$\sum_{e=1}^{N} (\psi_A^{(e)} \boldsymbol{e}_k, \boldsymbol{h}^{(e)}) := \sum_{e=1}^{N} \int_{\bar{\Omega}^e} \mathcal{G}(\psi_A^{(e)} \boldsymbol{e}_k) d\Omega = (\psi_A \boldsymbol{e}_k, \boldsymbol{h}) \tag{5.10}$$

is defined.

It is possible, through the use of an appropriate index renumbering operator, to write (5.9) and (5.10) in vector form. This operator $\mathrm{ID}(i, A)$, maps a pair of nodal and spatial indices A and i respectively, to a unique number. Thus (5.9) and (5.10) can be written as

$$\boldsymbol{K} = [K_P] = \sum^{N} \boldsymbol{k}^{(e)} \qquad \boldsymbol{F} = [F_P] = \sum^{N} \boldsymbol{h}^{(e)}$$

$$\boldsymbol{k}^{(e)} = k_P^{(e)} = \sum_{l=1}^{n\_dim} \sum_{B} a^{(e)} (\psi_A^{(e)} \boldsymbol{e}_k, \alpha_{Bl} \psi_B^{(e)} \boldsymbol{e}_l) \qquad \boldsymbol{h}^{(e)} = f_P^{(e)} = (\psi_A^{(e)} \boldsymbol{e}_k, \boldsymbol{h}^{(e)})$$

$$P = \mathrm{ID}(k, A)$$

An important result is that due to (5.7), (5.9) and (5.10) will only evaluate to non-zero if $\boldsymbol{x}_A, \boldsymbol{x}_B \in \bar{\Omega}^e$. This implies that $\boldsymbol{K}$ is sparce.

5.5 The approximate solution

We have successfully constructed a basis for $\mathcal{V}^h$, given by the function described in the previous section. Thus any $\boldsymbol{v} \in \mathcal{V}^h$ can be represented by

$$\boldsymbol{v}(\boldsymbol{x}) = \sum_{A=1} \sum_{j=1}^{n\_dim} \alpha_{Aj} \psi_A(\boldsymbol{x}) \boldsymbol{e}_j. \tag{5.11}$$

This applies to $\boldsymbol{v}^h$, $\boldsymbol{g}^h$ and $\boldsymbol{w}^h$ in (5.4). Applying this to $\boldsymbol{v}^h$, $\boldsymbol{g}^h$ and $\boldsymbol{w}^h$ we get

$$\boldsymbol{v}^h(\boldsymbol{x}) = \sum_{A \in \upsilon/\upsilon_g} \sum_{j}^{n\_dim} d_{Aj} \psi_A(\boldsymbol{x}) \boldsymbol{e}_j, \tag{5.12}$$

$$\boldsymbol{g}^h(\boldsymbol{x}) = \sum_{A \in \upsilon_g} \sum_{j}^{n\_dim} \beta_{Aj} \psi_A(\boldsymbol{x}) \boldsymbol{e}_j, \tag{5.13}$$

$$\boldsymbol{w}^h(\boldsymbol{x}) = \sum_{A \in \upsilon/\upsilon_g} \sum_{j}^{n\_dim} \gamma_{Aj} \psi_A(\boldsymbol{x}) \boldsymbol{e}_j. \tag{5.14}$$

Substituting these functions back into the Garlerkin formulation and factorising, we obtain

$$\sum_{l=1}^{n\_dim} \sum_{B \in \upsilon/\upsilon_g} a(\psi_A \boldsymbol{e}_k, d_{Bl} \psi_B \boldsymbol{e}_l) = (\psi_A \boldsymbol{e}_k, \boldsymbol{b}) + (\psi_A \boldsymbol{e}_k, \boldsymbol{h})_\Gamma - \sum_{l=1}^{n\_dim} \sum_{B \in \upsilon_g} a(\psi_A \boldsymbol{e}_k, \beta_{Bl} \psi_B \boldsymbol{e}_l). \tag{5.15}$$

The β_{Bl} can easily be determined since $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{g}^h(\boldsymbol{x})$ when $\boldsymbol{x}_B \in \Gamma_D$; that is,

$$\boldsymbol{g}(\boldsymbol{x}_B) = \sum_{A \in \upsilon_g} \sum_{j}^{n\_dim} \beta_{Aj} \psi_A(\boldsymbol{x}_B) \boldsymbol{e}_j, \tag{5.16}$$

$$\boldsymbol{g}(\boldsymbol{x}_B) = \sum_{j}^{n\_dim} \beta_{Bj} \psi_B(\boldsymbol{x}_B) \boldsymbol{e}_j, \tag{5.17}$$

$$\boldsymbol{g}(\boldsymbol{x}_B) = \sum_{j}^{n\_dim} \beta_{Bj} \boldsymbol{e}_j, \tag{5.18}$$

$$g_j(\boldsymbol{x}_B) = \beta_{Bj}. \tag{5.19}$$

A similar process can be applied to the approximation of the displacement $\boldsymbol{u}^h = \boldsymbol{v}^h + \boldsymbol{g}^h$ when considering a node $\boldsymbol{x}_B \notin \Gamma_D$. This results is $u_l^h = d_{Bl}$. Thus, d_{Bl} are interpreted directly as the displacements associated with nodes $\boldsymbol{x}_B \notin \Gamma_D$.

Now the primary objective is to determine the d_{Bl} in problem (5.15). Using the results from (5.9) and (5.10), and in particular how we got their vector representation, we can write (5.15) as

$$\boldsymbol{F}^{\text{int}} = \boldsymbol{F}^{\text{ext}} \tag{5.20a}$$

where

$$\left[F_P^{\text{int}}\right] = \sum^{N} \boldsymbol{k}^{(e)} \tag{5.20b}$$

$$\left[F_P^{\text{ext}}\right] = \sum^{m} \boldsymbol{h}^{(e)} \tag{5.20c}$$

$$\left[k_P^{(e)}\right] = \sum_{l=1}^{n\_dim} \sum_{B \in \upsilon/\upsilon_g} a^{(e)}(\psi_A^{(e)} \boldsymbol{e}_k, d_{Bl} \psi_B^{(e)} \boldsymbol{e}_l) \tag{5.20d}$$

$$\left[h_P^{(e)}\right] = (\psi_A^{(e)} \boldsymbol{e}_k, \boldsymbol{b}^{(e)}) + (\psi_A^{(e)} \boldsymbol{e}_k, \boldsymbol{h}^{(e)})_\Gamma - \sum_{l=1}^{n\_dim} \sum_{B \in \upsilon_g} a(\psi_A^{(e)} \boldsymbol{e}_k, \beta_{Bl} \psi_B^{(e)} \boldsymbol{e}_l) \tag{5.20e}$$

$$P = \text{ID}(k, A) \tag{5.20f}$$

If the stress-strain relationship is linear, as it is in linear elasticity, then (5.15) can be written as

$$\sum_{l=1}^{n\_dim} \sum_{B \in \upsilon/\upsilon_g} a(\psi_A \boldsymbol{e}_k, \psi_B \boldsymbol{e}_l) d_{Bl} = (\psi_A \boldsymbol{e}_k, \boldsymbol{b}) + (\psi_A \boldsymbol{e}_k, \boldsymbol{h})_\Gamma - \sum_{l=1}^{n\_dim} \sum_{B \in \upsilon_g} a(\psi_A \boldsymbol{e}_k, \beta_{Bl} \psi_B \boldsymbol{e}_l).$$

In this case the problem can be written as a matrix-vector form

$$\boldsymbol{K}\boldsymbol{d} = \boldsymbol{F},$$

where

$$[K_{PQ}] = \sum^{N} a^{(e)}(\psi_A \boldsymbol{e}_k, \psi_B \boldsymbol{e}_l)$$
$$[F_P] = \sum^{N} (\psi_A^{(e)} \boldsymbol{e}_k, \boldsymbol{b}^{(e)}) + (\psi_A^{(e)} \boldsymbol{e}_k, \boldsymbol{h}^{(e)})_\Gamma - \sum_{l=1}^{n\_dim} \sum_{B \in v_g} a(\psi_A^{(e)} \boldsymbol{e}_k, \psi_B^{(e)} \boldsymbol{e}_l)\beta_{Bl}$$
$$P = \mathrm{ID}(k, A) \qquad Q = \mathrm{ID}(l, B).$$

5.6 Solving the system of governing equations

Now we will give a brief discussion of how to solve equation (5.20a). Since the applied forces are time-dependent, the problem and its variables are time-dependent. We write (5.20a) in terms of its independent variables as

$$\boldsymbol{F}^{\mathrm{int}}(\boldsymbol{\sigma}) - \boldsymbol{F}^{\mathrm{ext}}(t) = \boldsymbol{0}. \tag{5.21}$$

We discretise time into increments $\Delta t = t_{n+1} - t_n$. Equation (5.21) must hold for each instant of time t_n. The time increament form of (5.21) is thus given by

$$\boldsymbol{F}^{\mathrm{int}}(\boldsymbol{\sigma}_n) - \boldsymbol{F}^{\mathrm{ext}}(t_n) = \boldsymbol{0}. \tag{5.22}$$

This results in the time-dependent applied forces (given by the boundary condition) being stated as incremental applied forces. Thus, $\boldsymbol{F}^{\mathrm{ext}}$ is known at every time step and denoted by $\boldsymbol{F}^{\mathrm{ext}}(t_n) := \boldsymbol{F}^{\mathrm{ext}}_n$. We assume that the displacements and the stress are known at some time t_n, such that (5.22) is satisfied. The value of the displacements and the stress at t_n are denoted by $\boldsymbol{d}_n$ and $\boldsymbol{\sigma}_n$ respectively. The increment in the displacemnet is denoted by $\Delta \boldsymbol{d}_n = \boldsymbol{d}_{n+1} - \boldsymbol{d}_n$. The question now remains to obtain $\Delta \boldsymbol{d}_n$ and $\boldsymbol{\sigma}_{n+1}$ such that

$$\boldsymbol{F}^{\mathrm{int}}(\boldsymbol{\sigma}_{n+1}) - \boldsymbol{F}^{\mathrm{ext}}_{n+1} = \boldsymbol{0}.$$

We linearise this problem and implement a Newton–Raphson scheme to solve for $\delta \boldsymbol{d}^k$, the Newton iteration of $\Delta \boldsymbol{d}_n$:

$$\underbrace{\left[\boldsymbol{F}^{int}(\boldsymbol{\sigma}^k_{n+1}) - \boldsymbol{F}^{\text{ext}}_{n+1}\right]}_{-\boldsymbol{R}} + \underbrace{\left[\frac{\partial \boldsymbol{F}^{int}(\boldsymbol{\sigma})}{\partial \boldsymbol{d}}\right]^k_{n+1}}_{\boldsymbol{J}^k} \delta \boldsymbol{d}^{k+1} = \boldsymbol{0} \tag{5.23}$$

$$\Rightarrow \begin{cases} \boldsymbol{J}^k \delta \boldsymbol{d}^{k+1} = \boldsymbol{R}, \\ \Delta \boldsymbol{d}_n = \sum_k \delta \boldsymbol{d}^k \\ \boldsymbol{d}^k_n = \sum_n \Delta \boldsymbol{d}_n \\ \boldsymbol{\sigma}^k_{n+1} = \mathcal{C}^e(\boldsymbol{\varepsilon}(\boldsymbol{d}^k_n) - \boldsymbol{\varepsilon}^p(\boldsymbol{d}^k_n)) \end{cases}$$

where the superscripts refer to increments in the Newton–Raphson scheme and subscripts refer to increments in time. The tangent $\partial \boldsymbol{F}^{int}/\partial \boldsymbol{d}$ is as follows: in index form

$$\begin{aligned} \frac{\partial F^{int}_P}{\partial d_Q} &= \sum_e \frac{\partial k^{(e)}_P}{\partial d_Q} \\ \frac{\partial k^{(e)}_P}{\partial d_Q} &= \frac{\partial}{\partial d_{Cj}} \sum_{l=1}^{n\_dim} \sum_{B \in \upsilon/\upsilon_g} a^{(e)}(\psi^{(e)}_A \boldsymbol{e}_k, d_{Bl}\psi^{(e)}_B \boldsymbol{e}_l) \\ &= \sum_{l=1}^{n\_dim} \sum_{B \in \upsilon/\upsilon_g} \frac{\partial}{\partial d_{Cj}} a^{(e)}(\psi^{(e)}_A \boldsymbol{e}_k, d_{Bl}\psi^{(e)}_B \boldsymbol{e}_l) \\ &= \sum_{l=1}^{n\_dim} \sum_{B \in \upsilon/\upsilon_g} \frac{\partial}{\partial d_{Cj}} \int_{\bar{\Omega}^e} \text{sym}(\nabla \psi^{(e)}_A \boldsymbol{e}_k) : \boldsymbol{\sigma} dx \\ &= \sum_{l=1}^{n\_dim} \sum_{B \in \upsilon/\upsilon_g} \int_{\bar{\Omega}^e} \text{sym}(\nabla \psi^{(e)}_A \boldsymbol{e}_k) : \frac{\partial \boldsymbol{\sigma}}{\partial d_{Cj}} dx \\ &= \sum_{l=1}^{n\_dim} \sum_{B \in \upsilon/\upsilon_g} \int_{\bar{\Omega}^e} \text{sym}(\nabla \psi^{(e)}_A \boldsymbol{e}_k) : \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}} : \frac{\partial \boldsymbol{\varepsilon}}{\partial d_{Cj}} dx \end{aligned}$$

(applying the chain rule)

$$= \sum_{l=1}^{n\_dim} \sum_{B \in v/v_g} \int_{\bar{\Omega}^e} \mathrm{sym}(\nabla \psi_A^{(e)} \boldsymbol{e}_k) : \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}} : \frac{\partial}{\partial d_{Cj}} \mathrm{sym}(\nabla d_{Dn} \psi_D^{(e)} \boldsymbol{e}_n) dx$$

$$= \sum_{l=1}^{n\_dim} \sum_{B \in v/v_g} \int_{v} \mathrm{sym}(\nabla \psi_A^{(e)} \boldsymbol{e}_k) : \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}} : \mathrm{sym}(\nabla \psi_C^{(e)} \boldsymbol{e}_j) dx. \tag{5.24}$$

The term $\partial \boldsymbol{\sigma} / \partial \boldsymbol{\varepsilon}$ requires an exact form of the stress relationship, and it is known as the *algorithmic tangent modulus*. Its time-discrete form is written as

$$\boldsymbol{C}_{n+1}^k := \frac{\partial \boldsymbol{\sigma}_{n+1}^k}{\partial \boldsymbol{\epsilon}_{n+1}^k}.$$

This will be the subject of discussion in a subsequent chapter.

This is by no means a complete discussion of finite elements. Numerous issues still remain. For instance, how to perform the integration in (5.20). In brief, one uses Gauss quadrature and isoparametric mapping. Given any function $g(\boldsymbol{x})$ that needs to be integrated over the element $\bar{\Omega}^e$, one uses isoparametric mapping to map the element to the *reference element* $\bar{\Omega}^r$. On the reference element, one can integrate the function using Gauss quadrature, with Gauss points and weights $\boldsymbol{\gamma}_i$ and ν_i respectively. Therefore,

$$\int_{\bar{\Omega}^e} g(\boldsymbol{x}) d\boldsymbol{x} = \int_{\bar{\Omega}^r} g(\boldsymbol{\xi}) \det(\boldsymbol{J}) d\boldsymbol{\xi} \approx \sum_i \nu_i g(\boldsymbol{\gamma}_i) \det(\boldsymbol{J}) \tag{5.25}$$

where $\boldsymbol{\xi} \in \bar{\Omega}^r$ and $\boldsymbol{J}$ is the Jacobian for the isoparametric mapping form $\bar{\Omega}^e$ to $\bar{\Omega}^r$.

The use of Gauss quadrature has an important implication for our problem as Simo and Hughes [39] point out. The implication is that that the stress within an element is required only at discrete points, typically the quadrature points of the element. This means that it is only necessary to calculate the stress at specific points. Later this will lead to additional algorithms that need to be solved at the quadrature points.

An outline of an algorithm for finite elements is given in Algorithm 1. We refer a single Newton–Raphson iteration of this algorithm as a *global Newton–Raphson iteration*.

Algorithm 1: A single time step of a finite element algorithm (a global Newton–Raphson)

Data: Boundary conditions for the time step: applied tractions $\boldsymbol{h}$; $\boldsymbol{u}$ on Γ_D; body forces $\boldsymbol{b}$. Data from previous time step: displacements $\boldsymbol{u}_n$; stress $\boldsymbol{\sigma}_n$; consistent tangent $\boldsymbol{C}_n$

Result: $\boldsymbol{u}_{n+1}$; $\boldsymbol{\sigma}_{n+1}$; $\boldsymbol{C}_{n+1}$

Calculate: d_{Bl} from $\boldsymbol{u}_n$ and β_{Bl} from $\boldsymbol{u}$ on Γ_D;

$k = 0$;

$\boldsymbol{\sigma}^k_{n+1} = \boldsymbol{\sigma}_n$; $\boldsymbol{C}^k_{n+1} = \boldsymbol{C}_n$;

(†)Assemble: $\boldsymbol{R} = \boldsymbol{F}^{\text{ext}} - \boldsymbol{F}^{int}(\boldsymbol{\sigma}^k_{n+1})$;

if $k > 0$ *and* $|\boldsymbol{R}| < tol$ **then**

| Move to next time step

end

Assemble: $\boldsymbol{J}$ using $\boldsymbol{C}^k_{n+1}$;

Solve: $\boldsymbol{J}\delta\boldsymbol{d} = \boldsymbol{R}$;

Update: $\Delta\boldsymbol{d} = \Delta\boldsymbol{d} + \delta\boldsymbol{d}$; $\boldsymbol{d}_{n+1} = \boldsymbol{d}_n + \Delta\boldsymbol{d}$;

Calculate $\boldsymbol{\sigma}^k_{n+1}$ and $\boldsymbol{C}^k_{n+1}$ at the quadrature points (this may require running additional algorithms);

$k = k + 1$;

return to (†);

Chapter 6

Algorithms for single crystal plasticity

6.1 Time and spatial discretisation

The time domain $[0, T]$ is subdivided into N equal intervals of duration $\Delta t = T/N$. The time at step n is denoted $t_n = n\Delta t$. We set $a(t_n) = a_n$, for any variable a. The complete state of the system is assumed known at t_n; as is the external loading at t_{n+1}. The objective is to determine the increment in total and plastic strain over the time step.

The discrete form of the yield function, obtained using (3.13) and (3.14), is

$$\phi^\alpha(\boldsymbol{\varepsilon}^p_{n+1}, \chi^\alpha_{n+1}) = \left[\mathcal{C}^e\left(\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}^p_{n+1}\right)\right] : \boldsymbol{P}^\alpha - \left(\tau^\alpha_0 - \chi^\alpha_{n+1}\right). \tag{6.1}$$

The discrete form of the hardening law is obtained by applying, for example, a backward Euler approximation to the hardening law (3.16): that is,

$$\chi^\alpha_{n+1}(\Delta\boldsymbol{\lambda}) = \chi^\alpha_n + \sum_\beta h^{\alpha\beta}(A_{n+1})\Delta\lambda^\beta, \tag{6.2}$$

where we represent the entire set of $\Delta\lambda^\alpha$ collectively as $\Delta\boldsymbol{\lambda}$ and $A_{n+1} = \sum_\alpha (\lambda^\alpha_n + \Delta\lambda^\alpha)$.

In a straightforward manner we apply the backward Euler method to the time derivatives

in equations (3.32), (3.34) and (3.35), resulting in the following set of flow equations:

$$\boldsymbol{\varepsilon}^p_{n+1}(\Delta\boldsymbol{\lambda}) = \boldsymbol{\varepsilon}^p_n + \sum_\alpha \Delta\lambda^\alpha \boldsymbol{P}^\alpha \tag{6.3a}$$

Lagrange multiplier [KKT] $$\phi^\alpha(\Delta\boldsymbol{\lambda}) \leq 0,\ \Delta\lambda^\alpha \geq 0,\ \phi^\alpha(\Delta\boldsymbol{\lambda})\Delta\lambda^\alpha = 0 \tag{6.3b}$$

Penalty $$\Delta\lambda^\alpha = \Delta t\eta \left[\left(\frac{\max[0, \phi^\alpha]}{\tau^\alpha_0 - \chi^\alpha} + 1 \right)^p - 1 \right] \tag{6.3c}$$

Augmented Lagrangian $$\Delta\lambda^\alpha = \max\left[0, \Delta\lambda^\alpha + \Delta t\eta\phi^\alpha(\boldsymbol{\varepsilon}^p_{n+1}, \chi^\alpha_{n+1})\right]$$

Equation (6.3a) and (6.2) only depends on $\Delta\lambda^\alpha$, which implies that the discrete yield condition, equation (6.1), can be reduced to a function of $\Delta\lambda^\alpha$:

$$\phi^\alpha(\Delta\boldsymbol{\lambda}) = \left[\mathcal{C}^e \left(\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}^p_n - \sum_\beta \Delta\lambda^\beta \boldsymbol{P}^\beta \right) \right] : \boldsymbol{P}^\alpha - \left(\tau^\alpha_0 - \chi^\alpha_n - \sum_\beta h^{\alpha\beta}(A_{n+1})\Delta\lambda^\beta \right).$$

The spatial domain is partitioned in a similar way to that described in Section 5.4. We employ a finite element approximation to the equilibrium equation. This results in the Newton–Raphson procedure described in (5.23) for the displacement. Here we need an expression for the stress. From the constitutive relationships for single crystal plasticity we can write the stress, using (3.13) and (3.14), as

$$\boldsymbol{\sigma}^k_{n+1} = \mathcal{C}^e(\boldsymbol{\varepsilon}^k_{n+1} - (\boldsymbol{\varepsilon}^p)^k_{n+1}),$$

where k is the Newton iteration counter. Using the discrete flow rule (6.3a) this becomes

$$\begin{aligned} \boldsymbol{\sigma}^k_{n+1} &= \mathcal{C}^e \left(\boldsymbol{\varepsilon}^k_{n+1} - \left(\boldsymbol{\varepsilon}^p_n + \sum_\alpha \Delta\lambda^\alpha \boldsymbol{P}^\alpha \right)^k \right) = \mathcal{C}^e \left(\boldsymbol{\varepsilon}^k_{n+1} - \boldsymbol{\varepsilon}^p_n - \sum_\alpha \Delta\lambda^\alpha \boldsymbol{P}^\alpha \right) \\ &= \mathcal{C}^e \left(\mathrm{sym}(\nabla \boldsymbol{u}^k_{n+1}) - \boldsymbol{\varepsilon}^p_n - \sum_\alpha \Delta\lambda^\alpha \boldsymbol{P}^\alpha \right) \end{aligned} \tag{6.4}$$

since $\boldsymbol{\varepsilon}^p_n$ and $\Delta\lambda^\alpha$ are in some sense independent of the global Newton–Raphson iteration, as we shall see in the following section.

6.2 Algorithm outline

This section present a strategy to solve the problem. We shall implement what is known as a *predictor-corrector* method for this problem. We introduce the trial elastic strain[1]

$$\boldsymbol{\varepsilon}^{e*} = \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_n^p = \mathrm{sym}(\nabla \boldsymbol{u}) - \boldsymbol{\varepsilon}_n^p. \tag{6.5}$$

Using $\boldsymbol{\varepsilon}^e = \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^p$ and substituting for $\boldsymbol{\varepsilon}^p$ from the flow law (6.3a), we get

$$\boldsymbol{\varepsilon}^e = \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^p = \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_n^p - \sum_\alpha \Delta\lambda^\alpha \boldsymbol{P}^\alpha,$$

which can be written using the trial elastic strain (6.5) as

$$\boldsymbol{\varepsilon}^e = \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^p = \boldsymbol{\varepsilon}^{e*} - \sum_\alpha \Delta\lambda^\alpha \boldsymbol{P}^\alpha.$$

Using the trial elastic strain we can calculate the trial stress, trial Schmid stress and the trial yield function as

$$\boldsymbol{\sigma}^* = \mathcal{C}^e \boldsymbol{\varepsilon}^{e*} \tag{6.6a}$$

$$\tau^{\alpha*} = \boldsymbol{\sigma}^* : \boldsymbol{P}^\alpha \tag{6.6b}$$

$$\phi^{\alpha*} = \tau^{\alpha*} - (\tau_0^\alpha - \chi_n^\alpha). \tag{6.6c}$$

At the beginning of every time step we assume that the first global Newton–Raphson iteration is elastic, i.e. $\Delta\lambda^\alpha = 0$ and $\Delta\chi_n^\alpha = 0$, and $\boldsymbol{\sigma} = \boldsymbol{\sigma}_n$. The Dirichlet boundary conditions are applied to the body and $\boldsymbol{F}_{n+1}^{ext}$ is assembled. Based on these assumptions we initiate a Newton step in (5.23) to determine the change in displacement $\delta\boldsymbol{u}$[1]. Using this we calculate the trial elastic strain $\boldsymbol{\varepsilon}^{e*}$ by (6.5) and the other trial quantities (6.6) at the quadrature points. One then considers $\phi^{\alpha*}$ at the each quadrature point. If $\phi^{\alpha*} < 0$ at a quadrature points, then for that quadrature point the global Newton–Raphson iteration is considered elastic with $\Delta\lambda^\alpha = 0$, $\Delta\chi_n^\alpha = 0$ and $\boldsymbol{\sigma} = \mathcal{C}^e\boldsymbol{\varepsilon}^{e*}$. If $\phi^{\alpha*} \geq 0$ at a quadrature points, then for that quadrature point plastic deformation potentially occurred over the global Newton–

[1]In what follows, all variables without subscripts are assumed to be evaluated at t_{n+1}.

Raphson iteration and the values of $\Delta\lambda^\alpha$, $\Delta\chi_n^\alpha$ and $\boldsymbol{\sigma} = \mathcal{C}^e(\boldsymbol{\varepsilon}^{e*} - \sum_\alpha \Delta\lambda^\alpha \boldsymbol{P}^\alpha)$ need to be determined. How this is done will be discussed later. Once this has been done for all quadrature points the next global Newton–Raphson iteration of (5.23) is initiated. We use this stress, and keep the assumptions $\Delta\lambda^\alpha = 0$ and $\Delta\chi_n^\alpha = 0$. This process is then repeated until convergence of the Newton–Raphson scheme (5.23) has been reached. The procedure is set out in Algorithm 2. The process of determining $\Delta\lambda^\alpha$ at a quadrature point is called a *local iteration*.

Algorithm 2: General solution scheme

Data: $\boldsymbol{\varepsilon}_n^p$, $\boldsymbol{u}_n$, χ_n^α, A_n
Result: $\boldsymbol{\varepsilon}_{n+1}^p$, $\boldsymbol{u}_{n+1}$, χ_{n+1}^α
Determine the stress from the previous time step $\boldsymbol{\sigma} = \mathcal{C}^e[\text{sym}(\nabla \boldsymbol{u}_n) - \boldsymbol{\varepsilon}_n^p]$.
(†) Solve one iteration of the Newton–Raphson scheme (5.23) to determine the $\delta\boldsymbol{u}^k$.
if *Newton–Raphson scheme* (5.23) *satisfied.* **then**
  $\boldsymbol{\varepsilon}_{n+1}^p = \boldsymbol{\varepsilon}_n^p + \sum_\alpha \Delta\lambda^\alpha \boldsymbol{P}^\alpha$
  $\chi_{n+1}^\alpha = \chi_n^\alpha + \Delta\chi_n^\alpha$.
  $A = A_n + \sum_\alpha \Delta\lambda^\alpha$ Start next time step.
end
Update $\boldsymbol{u}_n^k$ and calculate $\boldsymbol{\varepsilon}^{e*}$ and other trial variable by (6.5) and (6.6) at the quadrature points.
if $\phi^{\alpha*} \geq 0$ *at some quadrature points* **then**
  Solve for $\Delta\lambda^\alpha$ and $\Delta\chi^\alpha$ at those quadrature points (This process depends on the choice on flow rule formulation. This will be discussed in a following section).
  Calculate resulting stress $\boldsymbol{\sigma} = \mathcal{C}^e[\boldsymbol{\varepsilon}^{e*} - \sum_\alpha \Delta\lambda^\alpha \boldsymbol{P}^\alpha]$.
end
else
  Set $\Delta\lambda^\alpha = 0$ and $\Delta\chi^\alpha = 0$ at those quadratures points.
  Calculate resulting stress $\boldsymbol{\sigma} = \mathcal{C}^e\boldsymbol{\varepsilon}^{e*}$.
end
Return the (†)

6.3 Calculating $\Delta\lambda_n^\alpha$ and $\Delta\chi_n^\alpha$

The objective of this section is to present the algorithms for the Lagrange multiplier, penalty, and augmented Lagrangian formulations. Two distinct methodologies, based upon the Lagrange multiplier method, are discussed here. The first is the set search algorithm (SSA), and the second is an equivalent formulation algorithm (EA) due to Schmidt-Baldassari [37].

The augmented Lagrangian algorithm (ALA) is based on a fixed point algorithm.

6.3.1 The Lagrange multiplier approach by the set search algorithm (SSA)

The SSA is discussed in an abstract context in [26]. We define the *active set* of slip systems as

$$\mathcal{A} := \{\alpha \in \mathcal{P} | \Delta\lambda^\alpha > 0\}\,. \tag{6.7}$$

We denote the collection of incremental slip that are an element of the active set as

$$\Delta\bar{\boldsymbol{\lambda}} := \{\Delta\lambda^\alpha | \alpha \in \mathcal{A}\}.$$

Now we can write the KKT constraints in equation (6.3b) in terms of the active set as

$$\phi^\alpha(\Delta\bar{\boldsymbol{\lambda}}) = 0 \quad \text{and} \quad \Delta\lambda^\alpha > 0 \quad \text{for} \quad \alpha \in \mathcal{A}, \tag{6.8a}$$

$$\phi^\alpha(\Delta\bar{\boldsymbol{\lambda}}) < 0 \quad \text{and} \quad \Delta\lambda^\alpha = 0 \quad \text{for} \quad \alpha \in \mathcal{P}/\mathcal{A}. \tag{6.8b}$$

If we can determine an active set that satisfies both equation (6.8a) and (6.8b), then we have solved the problem. A solution algorithm is as follows:

1. Assume an active set.
2. Solve the equation $\phi^\alpha(\Delta\bar{\boldsymbol{\lambda}}) = 0$ for $\alpha \in \mathcal{A}$, using a Newton–Raphson scheme, see Section 4.3.
3. Check that $\phi^\alpha(\Delta\bar{\boldsymbol{\lambda}}) < 0$ for $\alpha \in \mathcal{P}/\mathcal{A}$ and $\Delta\lambda^\alpha > 0$ for $\alpha \in \mathcal{A}$. If one of these requirements is violated, update $\mathcal{A}$ and start again.

This approach has two fundamental difficulties. Firstly the Jacobian, required to solve $\phi^\alpha(\Delta\bar{\boldsymbol{\lambda}}) = 0$ for $\alpha \in \mathcal{A}$ in the Newton–Raphson formulation, defined as

$$K^{\alpha\beta} = \frac{\partial\phi^\alpha\left(\Delta\bar{\boldsymbol{\lambda}}\right)}{\partial\Delta\lambda^\beta} = -\boldsymbol{P}^\alpha : \mathcal{C}^e\boldsymbol{P}^\beta + \sum_{\delta\in\mathcal{A}}\left(\frac{\partial h^{\alpha\beta}}{\partial A}\Delta\lambda^\delta + h^{\alpha\beta}\delta^{\delta\beta}\right) \quad \alpha,\,\beta \in \mathcal{A}, \tag{6.9}$$

has the potential to be singular or ill-conditioned. Secondly, the active set is not known *a priori*, so a heuristic algorithm for determining the active set must be used. One such algorithm, presented in [29], is summarised in Algorithm 3.

6.3.2 Source of the singular system

We first assume all the slip systems are independent, which is not always the case see [19, p. 50]. It is clear from equation (3.32a) that the set $\{\boldsymbol{P}^\alpha\}$ form a basis for $\dot{\boldsymbol{\varepsilon}}^p$. The symmetry and incompressibility[2] requirements on $\boldsymbol{\varepsilon}^p$ also apply to $\dot{\boldsymbol{\varepsilon}}^p$, implying that there are only five independent entries in $\dot{\boldsymbol{\varepsilon}}^p$. The case when the proposed active set $\mathcal{A}$ contains more than five slip systems presents a problem. This is because we are trying to find the scalars (the proposed $\Delta\bar{\boldsymbol{\lambda}}$), of a now overdetermined problem. This will manifest in a singular Jacobian. Of course, for the case where the slip systems are not independent the problem is then compounded. A solution to problems of this kind can be solved using singular value decomposition (SVD), see Section 4.2. As we shall see the viscoplastic approach and the ALA do not suffer from this issue as they solve a regularised problem.

This independence problem also causes issues with the set search. To illustrate: imagine a problem where there are only three slip systems, and slip system $\alpha = 1$ is a linear combination of $\alpha = 2$ and $\alpha = 3$. For some set of loading condition the active set $\mathcal{A} = \{1\}$ is a solution to the set search problem. Since $\alpha = 1$ can be made of a linear combination of $\alpha = 2$ and $\alpha = 3$, this means for the same set of loading conditions the active set $\mathcal{A} = \{2, 3\}$ as well as the active set $\mathcal{A} = \{1, 2, 3\}$ are also solutions to the active set search problem. As a result, the process that we use to find the active set (or more specifically update the active set) is critical in representing the correct behaviour of the system.

6.3.3 The Lagrange multiplier approach by the "equivalent" formulation algorithm (EA).

Another approach, presented by [37], is to replace the KKT constraints (6.3b) with[3]

$$\sqrt{\left(\phi^\alpha\left(\Delta\boldsymbol{\lambda}\right)\right)^2 + \left(\Delta\lambda^\alpha\right)^2} + \phi^\alpha\left(\Delta\boldsymbol{\lambda}\right) - \Delta\lambda^\alpha = 0, \quad \forall\alpha \in \mathcal{P}\,. \tag{6.10}$$

It should be emphasise that the set of inequlity and equality constrains in (6.3b) have been replaced by an equation.

Equation (6.10) can be shown to be equivalent to the KKT constraints. By squaring the

[2] see (3.8).

[3] We omit the subscript $n + 1$ for convenience.

Algorithm 3: Set search algorithm (SSA)

Data: $\boldsymbol{\varepsilon}_n^p$, A_n, χ_n^α, $\mathcal{A}_n$
Result: $\boldsymbol{\varepsilon}^p$, $\Delta\lambda^\alpha$
$i = 1$, $\mathcal{A} = \mathcal{A}_n$
(†) $\Delta\lambda^\alpha = 0$
if $i = 2$ **then**
  $\mathcal{A} = \emptyset$
end
$i = i + 1$
while *First iteration or* $\max M^\alpha < tol$ **do**
  Construct residual for $\alpha \in \mathcal{A}$:
  $M^\alpha = \phi^\alpha\left(\Delta\bar{\boldsymbol{\lambda}}\right) = \left[\mathcal{C}^e\left(\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}_n^p - \sum_{\delta\in\mathcal{A}}\Delta\lambda^\delta \boldsymbol{P}^\delta\right)\right] : \boldsymbol{P}^\alpha - \left(\tau_0^\alpha - \chi^\alpha\left(\Delta\bar{\boldsymbol{\lambda}}\right)\right)$
  Construct Jacobian for $\alpha, \beta \in \mathcal{A}$:
  $K^{\alpha\beta} = -\boldsymbol{P}^\alpha : \mathcal{C}^e \boldsymbol{P}^\beta + \sum_{\delta\in\mathcal{A}}\left(\frac{\partial h^{\alpha\beta}}{\partial A}\Delta\lambda^\delta + h^{\alpha\beta}\delta^{\delta\beta}\right)$
  Update: $\Delta\lambda^\alpha = \Delta\lambda^\alpha - \left(K^{\alpha\beta}\right)^{-1} M^\alpha$, if $K^{\alpha\beta}$ is ill-conditioned use SVD.
    $A = A_n + \sum_\alpha \Delta\lambda^\alpha$
    $\chi^\alpha = \chi_n^\alpha + \sum_\beta h^{\alpha\beta}(A)\Delta\lambda^\beta$
end
if *for some* $\alpha \in \mathcal{A}$*:* $\Delta\lambda^\alpha < 0$ **then**
  $\mathcal{A} \Leftarrow \mathcal{A}/\{\alpha = \arg[\min \Delta\lambda^\alpha], \alpha \in \mathcal{A}\}$ and return to (†)
end
if *for some* $\alpha \in \mathcal{P}/\mathcal{A}$*:* $\phi^\alpha > 0$ **then**
  $\mathcal{A} \Leftarrow \mathcal{A} \cup \{\alpha | \phi^\alpha > 0, \alpha \in \mathcal{P}/\mathcal{A}\}$ and return to (†)
end

equation, we find

$$\phi^{\alpha}\left(\Delta\boldsymbol{\lambda}\right)\Delta\lambda^{\alpha}=0. \tag{6.11}$$

This implies that equation (6.10) satisfies the third KKT condition. If we assume $\phi^{\alpha}\left(\Delta\boldsymbol{\lambda}\right)$ is non-zero then equation (6.11) implies $\Delta\lambda^{\alpha}=0$. Now, using this in equation (6.10) we get

$$\sqrt{\phi^{\alpha}\left(\Delta\boldsymbol{\lambda}\right)^{2}}+\phi^{\alpha}\left(\Delta\boldsymbol{\lambda}\right)=0$$

or

$$|\phi^{\alpha}\left(\Delta\boldsymbol{\lambda}\right)|+\phi^{\alpha}\left(\Delta\boldsymbol{\lambda}\right))=0.$$

This is only true if,

$$\phi^{\alpha}\left(\Delta\boldsymbol{\lambda}\right)\leq 0 \tag{6.12}$$

which implies that (6.10) satisfies the second KKT condition. If we assume $\Delta\lambda^{\alpha}$ is non-zero then (6.11) implies $\phi^{\alpha}\left(\Delta\boldsymbol{\lambda}\right)=0$. Now using this in equation(6.10) we get

$$\sqrt{(\Delta\lambda^{\alpha})^{2}}-\Delta\lambda^{\alpha}=0$$

or

$$|\Delta\lambda^{\alpha}|-\Delta\lambda^{\alpha}=0.$$

This is only true if

$$\Delta\lambda^{\alpha}\geq 0. \tag{6.13}$$

This implies that (6.10) satisfies the first KKT condition. Thus, (6.10) serves as a replacement for the KKT condition.

This equation is valid for all slip systems $\alpha\in\mathcal{P}$, irrespective of their "activity". By solving equation (6.10), we will determine the incremental slip $\Delta\lambda^{\alpha}$ and the active set.

The active set is determined simply by satisfying equation (6.10), since it will be given by $\mathcal{A} = \{\alpha | \Delta\lambda^\alpha > 0\}$.

Equation (6.10) can be solved using a Newton–Raphson strategy, see Section 4.3, where the Jacobian is given by

$$K^{\alpha\beta} = \delta_{\alpha\beta}\left(\frac{\Delta\lambda^\alpha}{\sqrt{\left(\phi^\alpha\left(\Delta\boldsymbol{\lambda}\right)\right)^2 + (\Delta\lambda^\alpha)^2}} - 1\right) + \frac{\partial\phi^\alpha(\Delta\boldsymbol{\lambda})}{\partial\Delta\lambda^\beta}\left(\frac{\phi^\alpha\left(\Delta\boldsymbol{\lambda}\right)}{\sqrt{\left(\phi^\alpha\left(\Delta\boldsymbol{\lambda}\right)\right)^2 + (\Delta\lambda^\alpha)^2}} + 1\right). \quad (6.14)$$

This alternative form obviates the need to determine an active set, but at the expense of solving a system of $2N$ equations once. In the case of a face-centered cubic crystal (f.c.c.) structure, this implies 24 (12 doubly defined slip systems) equations. It can be shown though geometric arguments that solving the same problem using the SSA will at worst involve solving a final system of 8 equations. However, one may solve a significant number of smaller systems in the active set iterations to get to the final system. An attractive quality of the EA is that it has a mathematical foundation while the set search process in the SSA is heuristic. In this formulation, the Jacobian retains the possibility of becoming singular or ill-conditioned and should be inverted using SVD (see Section 4.2) or similar techniques. An algorithm for this formulation is summarised in Algorithm 4.

6.3.4 Penalty method algorithm for the viscoplastic formulation

The algorithm presented here is very similar to that presented in the SSA, Section 6.3.1. We define the *active set* of slip systems as

$$\mathcal{A} := \{\alpha \in \mathcal{P} | \phi^\alpha > 0\}.$$

We denote the collection of incremental slips that are an element of the active set as

$$\Delta\bar{\boldsymbol{\lambda}} := \{\Delta\lambda^\alpha | \alpha \in \mathcal{A}\}.$$

Algorithm 4: Equivalent formulation algorithm (EA)

Data: $\boldsymbol{\varepsilon}_n^p$, A_n, χ_n^α
Result: $\boldsymbol{\varepsilon}^p$, $\Delta\lambda^\alpha$
$\Delta\lambda^\alpha = 0$;
Construct residual:
$M^\alpha = \sqrt{\left(\phi^\alpha\left(\Delta\boldsymbol{\lambda}\right)\right)^2 + \left(\Delta\lambda^\alpha\right)^2} + \phi^\alpha\left(\Delta\boldsymbol{\lambda}\right) - \Delta\lambda^\alpha$
Construct Jacobian:

$$K^{\alpha\beta} = \delta_{\alpha\beta}\left(\frac{\Delta\lambda^\alpha}{\sqrt{\left(\phi^\alpha\left(\Delta\boldsymbol{\lambda}\right)\right)^2 + \left(\Delta\lambda^\alpha\right)^2}} - 1\right) + \frac{\partial\phi^\alpha(\Delta\boldsymbol{\lambda})}{\partial\Delta\lambda^\beta}\left(\frac{\phi^\alpha\left(\Delta\boldsymbol{\lambda}\right)}{\sqrt{\left(\phi^\alpha\left(\Delta\boldsymbol{\lambda}\right)\right)^2 + \left(\Delta\lambda^\alpha\right)^2}} + 1\right)$$

Update: $\Delta\lambda^\alpha = \Delta\lambda^\alpha - \left(K^{\alpha\beta}\right)^{-1} M^\alpha$, if $K^{\alpha\beta}$ is ill-conditioned use SVD.
$A = A_n + \sum_\alpha \Delta\lambda^\alpha$
$\chi^\alpha = \chi_n^\alpha + \sum_\beta h^{\alpha\beta}(A)\Delta\lambda^\beta$

Using equation (6.3c) we can conclude that

$$\Delta\lambda^\alpha > 0 \quad \text{for} \quad \alpha \in \mathcal{A}, \tag{6.15a}$$

$$\phi^\alpha(\Delta\bar{\boldsymbol{\lambda}}) < 0 \quad \text{for} \quad \alpha \in \mathcal{P}/\mathcal{A}. \tag{6.15b}$$

If we can determine an active set that satisfies both equation (6.15a) and equation (6.15b), then we have solved the problem. In the case of the viscoplastic formulation, see (3.33), we can now write equation (6.3c) as

$$\Delta\lambda^\alpha = \begin{cases} \Delta t\eta\left[\left(\frac{\phi^\alpha}{\tau_0^\alpha - \chi^\alpha} + 1\right)^p - 1\right] & \alpha \in \mathcal{A}, \\ 0 & \alpha \in \mathcal{P}/\mathcal{A}. \end{cases}$$

Given an active set this can be written as

$$M^\alpha = \boldsymbol{\sigma} : \boldsymbol{P}^\alpha - (\tau_0^\alpha - \chi^\alpha)\left(\frac{\Delta\lambda^\alpha}{\Delta t\eta} + 1\right)^{1/p} = 0, \quad \alpha \in \mathcal{A}, \tag{6.16}$$

$$\Delta\lambda^\alpha = 0, \quad \alpha \notin \mathcal{A}.$$

A solution algorithm for the viscoplastic formulation is as follows:

1. Assume an active set.
2. Set $\Delta\lambda^\alpha = 0$ for all $\alpha \in \mathcal{P}/\mathcal{A}$.

3. Solve the equation $\boldsymbol{\sigma} : \boldsymbol{P}^\alpha - (\tau_0^\alpha - \chi^\alpha)\left(\frac{\Delta\lambda^\alpha}{\Delta t\eta} + 1\right)^{1/p} = 0$, using a Newton–Raphson scheme, see Section 4.3.

4. Check that $\phi^\alpha(\Delta\bar{\boldsymbol{\lambda}}) < 0$ for $\alpha \in \mathcal{P}/\mathcal{A}$ and $\Delta\lambda^\alpha > 0$ for $\alpha \in \mathcal{A}$. If at least one of these requirements is violated, update $\mathcal{A}$ and start again.

Unlike the SSA, the penalty method algorithm does not suffer from the difficulties associated with a singular problem. This is because the penalty method does not enforce a solution that satisfies the normality conditions exactly. As stated, for the viscoplastic formulation one is required to solve (6.16) using a Newton–Raphson method, where the Jacobian is given by

$$K^{\alpha\beta} = -\boldsymbol{P}^\alpha : \mathcal{C}^e \boldsymbol{P}^\beta + \left[\sum_{\delta\in\mathcal{A}}\left(\frac{\partial h^{\alpha\beta}}{\partial A}\Delta\lambda^\delta + h^{\alpha\beta}\delta^{\delta\beta}\right)\right]\left(\frac{\Delta\lambda^\alpha}{\Delta t\eta} + 1\right)^{1/p} - (\tau_0^\alpha - \chi^\alpha)\delta^{\alpha\beta}\frac{1}{p\eta\Delta t}\left(\frac{\Delta\lambda^\alpha}{\eta\Delta t} + 1\right)^{(1/p)-1} \tag{6.17}$$

An outline of the viscoplastic algorithm (VA) is given in Algorithm 5.

6.3.5 Fixed point algorithm for the augmented Lagrangian algorithm (ALA).

For this algorithm one can state the relevant set of equations as:

$$\begin{aligned}
\Delta\lambda^\alpha &= \max\left[0, \Delta\lambda^\alpha + \Delta t\eta\phi^\alpha(\boldsymbol{\varepsilon}^p, \chi^\alpha)\right] \\
\boldsymbol{\varepsilon}^p &= \boldsymbol{\varepsilon}_n^p + \sum_\alpha \max\left[0, \Delta\lambda^\alpha + \Delta t\eta\phi^\alpha(\boldsymbol{\varepsilon}^p, \chi^\alpha)\right]\boldsymbol{P}^\alpha \\
A &= A_n + \sum_\alpha \max\left[0, \Delta\lambda^\alpha + \Delta t\eta\phi^\alpha(\boldsymbol{\varepsilon}^p, \chi^\alpha)\right] \\
\chi^\alpha &= \chi_n^\alpha + \sum_\beta h^{\alpha\beta}(A)\max\left[0, \Delta\lambda^\beta + \Delta t\eta\phi^\beta(\boldsymbol{\varepsilon}^p, \chi^\beta)\right].
\end{aligned}$$

One can apply a fixed point approach to the equation for $\Delta\lambda^\alpha$; that is,

$$\Delta\lambda^\alpha_{(i+1)} = \max\left[0, \Delta\lambda^\alpha_{(i)} + \Delta t\eta\phi^\alpha(\boldsymbol{\varepsilon}^p_{(i)}, \chi^\alpha_{(i)})\right] \quad \text{with} \quad \Delta\lambda^\alpha_{(1)} = 0\,, \tag{6.18}$$

Algorithm 5: Viscoplastic algorithm (VA)

Data: ε_n^p, A_n, χ_n^α, $\mathcal{A}_n$
Result: ε^p, $\Delta\lambda^\alpha$
$i = 1$, $\mathcal{A} = \mathcal{A}_n$
(†) $\Delta\lambda^\alpha = 0$ for all $\alpha \in \mathcal{P}$
if $i = 2$ **then**
  $\mathcal{A} = \emptyset$
end
$i = i + 1$
while *First iteration or* $\max M^\alpha < tol$ **do**
  Construct residual for $\alpha \in \mathcal{A}$:
  $M^\alpha = \boldsymbol{\sigma} : \boldsymbol{P}^\alpha - (\tau_0^\alpha - \chi^\alpha)\left(\frac{\Delta\lambda^\alpha}{\Delta t\eta} + 1\right)^{1/p}$
  Construct Jacobian $\alpha, \beta \in \mathcal{A}$:

$$K^{\alpha\beta} = -\boldsymbol{P}^\alpha : \mathcal{C}^e \boldsymbol{P}^\beta + \sum_{\delta\in\mathcal{A}} \left(\frac{\partial h^{\alpha\beta}}{\partial A}\Delta\lambda^\delta + h^{\alpha\beta}\delta^{\delta\beta}\right)\left(\frac{\Delta\lambda^\alpha}{\Delta t\eta} + 1\right)^{1/p} - (\tau_0^\alpha - \chi^\alpha)\delta^{\alpha\beta}\frac{1}{p\eta\Delta t}\left(\frac{\Delta\lambda^\alpha}{\eta\Delta t} + 1\right)^{(1/p)-1}$$

  Update: $\Delta\lambda^\alpha = \Delta\lambda^\alpha - \left(K^{\alpha\beta}\right)^{-1} M^\alpha$, if $K^{\alpha\beta}$.
    $A = A_n + \sum_\alpha \Delta\lambda^\alpha$
    $\chi^\alpha = \chi_n^\alpha + \sum_\beta h^{\alpha\beta}(A)\Delta\lambda^\beta$
end
if *for some* $\alpha \in \mathcal{A}$*:* $\Delta\lambda^\alpha < 0$ **then**
  $\mathcal{A} \Leftarrow \mathcal{A}/\{\alpha = \arg[\min \Delta\lambda^\alpha], \alpha \in \mathcal{A}\}$ and return to (†)
end
if *for some* $\alpha \in \mathcal{P}/\mathcal{A}$*:* $\phi^\alpha > 0$ **then**
  $\mathcal{A} \Leftarrow \mathcal{A} \cup \{\alpha | \phi^\alpha > 0, \alpha \in \mathcal{P}/\mathcal{A}\}$ and return to (†)
end

where the iteration counter i denotes iterations within the fixed point algorithm at time step $n+1$. The intermediate states of the plastic strain $\boldsymbol{\varepsilon}^p_{(i)}$, accumulated slip $A_{(i)}$ and hardening $\chi^\alpha_{(i)}$ are given implicitly as

$$\begin{aligned}
\boldsymbol{\varepsilon}^p_{(i)}(\Delta\boldsymbol{\lambda}_{(i)}) &= \boldsymbol{\varepsilon}^p_n + \sum_\alpha \max\left[0, \Delta\lambda^\alpha_{(i)} + \Delta t\eta\phi^\alpha(\boldsymbol{\varepsilon}^p_{(i)}, \chi^\alpha_{(i)})\right]\boldsymbol{P}^\alpha,\\
A_{(i)}(\Delta\boldsymbol{\lambda}_{(i)}) &= A_n + \sum_\alpha \max\left[0, \Delta\lambda^\alpha_{(i)} + \Delta t\eta\phi^\alpha(\boldsymbol{\varepsilon}^p_{(i)}, \chi^\alpha_{(i)})\right],\\
\chi^\alpha_{(i)}(\Delta\boldsymbol{\lambda}_{(i)}) &= \chi^\alpha_n + \sum_\beta h^{\alpha\beta}\left(A_{(i)}\right)\max\left[0, \Delta\lambda^\beta_{(i)} + \Delta t\eta\phi^\beta(\boldsymbol{\varepsilon}^p_{(i)}, \chi^\beta_{(i)})\right].
\end{aligned}$$

Using (6.18) we can write the intermediate states as

$$\begin{aligned}
\boldsymbol{\varepsilon}^p_{(i)}(\Delta\boldsymbol{\lambda}_{(i+1)}) &= \boldsymbol{\varepsilon}^p_n + \sum_\alpha \Delta\lambda^\alpha_{(i+1)}\boldsymbol{P}^\alpha,\\
A_{(i)}(\Delta\boldsymbol{\lambda}_{(i+1)}) &= A_n + \sum_\alpha \Delta\lambda^\alpha_{(i+1)},\\
\chi^\alpha_{(i)}(\Delta\boldsymbol{\lambda}_{(i+1)}) &= \chi^\alpha_n + \sum_\beta h^{\alpha\beta}\left(A_{(i)}\right)\Delta\lambda^\beta_{(i+1)},
\end{aligned}$$

eliminating equation (6.18) dependence on $\boldsymbol{\varepsilon}^p_{(i)}$ and $\chi^\alpha_{(i)}$. This gives the reduced iterative scheme

$$\Delta\lambda^\alpha_{(i+1)} = \max\left[0, \Delta\lambda^\alpha_{(i)} + \Delta t\eta\phi^\alpha(\Delta\boldsymbol{\lambda}_{(i+1)})\right] \quad \text{with} \quad \Delta\lambda^\alpha_{(1)} = 0\,. \tag{6.20}$$

Equation (3.36) can be written as the time-discrete expression

$$\mathcal{A} = \{\alpha \in \mathcal{P} | \max\left[0, \Delta\lambda^\alpha + \Delta t\eta\phi^\alpha\right] \neq 0\}\,.$$

The state of the active set can be determined at any stage of the fixed point algorithm by

$$\mathcal{A}_{(i)} = \left\{\alpha \in \mathcal{P} | \max\left[0, \Delta\lambda^\alpha_{(i)} + \Delta t\eta\phi^\alpha(\Delta\lambda^\alpha_{(i)})\right] \neq 0\right\}. \tag{6.21}$$

As apposed to the SSA, this plays little role in the actual algorithm and the active set can be determined by a function evaluation. The active set can be monitored at any iteration of the fixed point algorithm. One may notice that the active set can only attain additional members as the fixed point algorithm progresses; this will be important later.

Note that the term $\Delta t\eta\phi^\alpha(\Delta\boldsymbol{\lambda}_{(i+1)})$ in equation (6.20) uses the update $\Delta\lambda^\alpha_{(i+1)}$. The fixed point algorithm will only converge if this term is sufficiently small - that is if $\phi^\alpha < \epsilon/\Delta t\eta$ for some tolerance ϵ. This implies that we have control over the satisfaction of the yield criterion through the choice of the penalty η. This formulation allows us to accelerate the convergence process by varying η. We adopt the strategy of increasing η between the fixed point steps:

$$\eta_{(i+1)} = m\eta_{(i)}, \quad \eta_{(1)} = \eta_0, \qquad m > 1\,. \tag{6.22}$$

This allows us to solve the first iteration of (6.20) with a small penalty factor η_0, thereby generating a well-conditioned problem (recall the interpretation of the penalty method as a form of viscoplastic regularisation). Thereafter, we increase η to allow for rapid convergence to the fixed point. We solve the fixed point problem (6.20) using a globally convergent Newton–Raphson method[4], with a residual defined as [5]

$$M^\alpha\left(\Delta\boldsymbol{\lambda}_{(i+1)}\right) = \Delta\lambda^\alpha_{(i+1)} - \max\left[0, \Delta\lambda^\alpha_{(i)} + \Delta t\eta_{(i)}\phi^\alpha(\Delta\lambda^\beta_{(i+1)})\right] \tag{6.23}$$

and Jacobian

$$\frac{\partial M^\alpha}{\partial\Delta\lambda^\beta_{(i+1)}} = \delta_{\alpha\beta} - \begin{cases} 0 \quad \text{if} \quad \max\left[0, \Delta\lambda^\alpha_{(i)} + \Delta t\eta_{(i)}\phi^\alpha\right] = 0, \\ \Delta t\eta_{(i)}\dfrac{\partial\phi^\alpha}{\partial\Delta\lambda^\beta_{(i+1)}} \quad \text{otherwise.} \end{cases} \tag{6.24}$$

The algorithm is summarised in Algorithm 6.

Why is it essential to use a globally convergent Newton–Raphson method for this problem? The Jacobian, equation (6.23), used in this algorithm has a nasty step-like behavior. The usual assumption of the Newton–Raphson method is that the function is convex and smooth in the vicinity of the root. However, in this case the Jacobian becomes ineffective (in a Newton–Raphson context) in the vicinity of the root, thus the quadratic assumption is invalid. If the Newton–Raphson process wanders into this region convergence will cease. A damped Newton–Raphson will assure that the Jacobian remains effective, and

[4] see Section 4.3.1

[5] We are solving for $\Delta\lambda^\alpha_{(i+1)}$ that satisfies the fixed point problem (6.20), so $\Delta\lambda^\alpha_{(i)}$ remains constant throughout the Newton–Raphson iterations.

Algorithm 6: Augmented Lagrangian algorithm (ALA)

Data: ε_n^p, A_n, χ_n^α
Result: ε^p, $\Delta\lambda^\alpha$
$\Delta\lambda^\alpha_{(1)} = 0$, $\eta_{(1)} = \eta_0$, $i = 1$
while $\max \phi^\alpha < tol$ **do**
  while *First iteration or* $\max M^\alpha < tol$ **do**
    Construct residual:
    $M^\alpha\left(\Delta\lambda^\beta_{(i+1)}\right) = \Delta\lambda^\alpha_{(i+1)} - \max\left[0, \Delta\lambda^\alpha_{(i)} + \Delta t \eta_{(i)} \phi^\alpha(\Delta\lambda^\beta_{(i+1)})\right]$
    Construct Jacobian:
    $K^{\alpha\beta} = \delta_{\alpha\beta} - \begin{cases} 0 & \text{if} \quad \max\left[0, \Delta\lambda^\alpha_{(i)} + \Delta t \eta_{(i)} \phi^\alpha\right] = 0 \\ \Delta t \eta_{(i)} \frac{\partial \phi^\alpha}{\partial \Delta\lambda^\beta_{(i+1)}} & \text{otherwise} \end{cases}$
    Update: $\Delta\lambda^\alpha_{(i+1)} = \Delta\lambda^\alpha_{(i+1)} - \left(K^{\alpha\beta}\right)^{-1} M^\alpha$
    $A = A_n + \sum_\alpha \Delta\lambda^\alpha_{(i+1)}$
    $\chi^\alpha = \chi^\alpha_n + \sum_\beta h^{\alpha\beta}(A)\Delta\lambda^\beta$
  end
  $\eta_{(i+1)} = m\eta_{(i)}$
  $i = i + 1$
end

ultimately converges to the root.

Due to the close vicinity of the root and the region where the Jacobian becomes ineffetctive, there is an unfortunate risk that the line search will dominate the root finding algorithm. This will maintain convergence, but have the undesirable effect of reducing the order of convergence.

6.3.6 Considering using an initial guess for $\Delta\lambda^\alpha$

All the algorithms presented thus far are based on the initial guess $\Delta\boldsymbol{\lambda} = \mathbf{0}$ at the start of the local algorithm. However, there appears to be no fundamental reason why this should be the case. For these algorithms the initial guess plays an important role, because central to each of these algorithms is a Newton–Raphson method. It is well-known that the convergence rate of the Newton–Raphson method is dependent on the initial guess [34, pg 362]. In this section we will consider alternative initial guesses for $\Delta\boldsymbol{\lambda}$ and discuss their effects.

An important consideration is whether the local problem is ill-conditioned. In the case that the local problem is not ill-conditioned, the problem has a unique solution. Any initial guess will ultimately lead to this solution, so when choosing an initial guess one would be concerned with the convergence rate. As we have discussed in Section 6.3.2, an ill-conditioned problem could have multiple solutions, and two different initial guesses could converge to two different solutions. It is important to note that from the perspective of the global algorithm that these two solutions are exactly the same, as they have the same stress state. Therefore, from the point of view of the global algorithm it seems fair to accept any solution provided by the local problem.

For many scenarios, the initial guess $\Delta\boldsymbol{\lambda} = \mathbf{0}$ seems an overly conservative approach. To illustrate, we shall consider the case of linear hardening with applied loading conditions that do not vary between time steps. Such loading conditions could be a constant applied displacement. Under these conditions, it is often the case that the solution for $\Delta\boldsymbol{\lambda}$ will have no variation between consecutive time steps. The only variation will occur at discrete times, when the active set changes. Taking this into consideration it is reasonable to expect that the initial guess $\Delta\boldsymbol{\lambda} = \Delta\boldsymbol{\lambda}_n$ would improve convergence. This approach does not apply only to the limited case we have considered above. If the harding rule is nonlinear, then the initial guess $\Delta\boldsymbol{\lambda} = \Delta\boldsymbol{\lambda}_n$ could be seen as a linear approximation of the behaviour. If the loading conditions are fairly varied, then it is fair to assume that the time steps would be small enough such that the boundary conditions appear constant. Using this initial guess is not as straightforward as one would like. We shall show that the initial guess comes with its own set of constraints.

An important consideration is whether the structure of the algorithm can effectively accommodate an initial guess. Consider for instance the SSA and the VA. These algorithms utilise a set search method which involves solving a series of problems, each building on the last attempt. The information of the current solution is used to propose a new active set, but that is all. No other information is transfered between consecutive active set steps. This is done for all set search iterations except the first, where an initial guess for the active set (i.e. $\mathcal{A} = \mathcal{A}_n$) is used. If the guess for the active set is not successful, the search restarts from an empty set. Therefore, if any initial guess for $\Delta\lambda^\alpha$ is proposed for this algorithm, it will only be effective in the first iteration. If it is not successful in the first iteration, the information is erased and the effect of the initial guess would be minimal in the greater scheme of the

local algorithm. So by their nature, set search methods do not lead themselves to an initial guess of $\Delta\lambda^\alpha$.

Now we consider algorithms that do not use an active set search approach. These are the ALA and the EA. These algorithms are not restricted by the set search process, and thus could benefit from the use of an initial guess. One would still have the problem associated with ill-conditioning and initial guess, but as we have stated there is no reason to claim that one solution is more correct than another. So, it seems that there is potential to use an initial guess $\Delta\lambda^\alpha = \Delta\lambda_n^\alpha$ for the ALA and the EA.

Another consideration is that the solution from time step to time step is not the only important state of the solution method. For each time step there are several intermediate global iterations, and associated with each is a local iteration. The converged local solution for one of these global steps could be different from the converged solution of the time step. This means that using an initial guess $\Delta\lambda^\alpha = \Delta\lambda_n^\alpha$ may not be entirely relevant for some of the intermediate local iterations. This acts as motivation for the algorithm to be flexible enough in moving from the initial guess to the solution, even if the initial guess is fairly different from the solution.

Basic precautions can be taken to ensure that the initial guess $\Delta\lambda^\alpha = \Delta\lambda_n^\alpha$ is beneficial. Firstly, an initial guess is only beneficial if it reduces the residual. So, before implementing the initial guess it would be wise to compare residuals of the two cases, $\Delta\lambda^\alpha = \Delta\lambda_n^\alpha$ and $\Delta\lambda^\alpha = 0$. Naturally, whichever leads to the smaller residual is used. Secondly, the initial guess is not used if $\phi^\alpha(\Delta\lambda_n^\alpha) < 0$ and $\Delta\lambda_n^\alpha > 0$ for some α.

In the following section we further discuss how the ALA can be used along with the initial guess $\Delta\lambda^\alpha = \Delta\lambda_n^\alpha$.

6.3.7 A modified algorithm

Here we shall discuss how we can modify the ALA solution scheme, using the initial guess $\Delta\lambda^\alpha = \Delta\lambda_n^\alpha$, such that the solution for $\Delta\lambda^\alpha$ is comparable to solution schemes that use the initial guess $\Delta\lambda^\alpha = 0$. There are scenarios where the initial guess $\Delta\lambda^\alpha = \Delta\lambda_n^\alpha$ could be seen to favour certain slip systems. Here we will propose a method for rectifying this. As we shall illustrate, this method is not foolproof, and there are conditions that will result in a

solution for $\Delta\lambda^\alpha$ that is different from solution schemes that use the initial guess $\Delta\lambda^\alpha = 0$.

This method addresses the situation where the time step is small and there are constant loading conditions. As we have stated, we generally expect the initial guess $\Delta\lambda^\alpha = \Delta\lambda_n^\alpha$ to be close to the solution. An implicit assumption when using this initial guess is that $\mathcal{A} = \mathcal{A}_n$, however this is not necessarily true. When this is not the case, and the slip systems being added to the active set are not a linear combination of those in $\mathcal{A}_n$, the initial guess is fine[6]. When the slip systems being added to the active set are a linear combination of those in $\mathcal{A}_n$, essentially that slip system is not necessarily needed, and the initial guess favours those in $\mathcal{A}_n$. In order to rectify this we decide to disregard the initial guess $\Delta\lambda^\alpha = \Delta\lambda_n^\alpha$, and use the initial guess $\Delta\lambda^\alpha = 0$. A prerequisite of this is an algorithm that has a robust definition of an active set. The ALA active set definition (6.21) meets this requirement. Its definition can be readily used at any step of the algorithm. One would try to use the initial guess $\Delta\lambda^\alpha = \Delta\lambda_n^\alpha$, then monitor the development of the active set throughout the local algorithm. If any slip system not in $\mathcal{A}_n$ wants to become active, the local algorithm is restarted with the initial guess $\Delta\lambda^\alpha = 0$. Using this strategy we can develop a modified algorithm. This is presented in Algorithm 7.

The problem with this modification is that it only uses the initial guess $\Delta\lambda^\alpha = 0$ if it detects a change in the active set. There are situations where you have multiple solutions within the same active set. In these situations, depending on the initial guess, one can converge onto different roots. This modification does not address this problem.

One may consider applying this approach to the EA. However, on a practical level this treatment does not have a robust definition of an active set. The active set can theoretically be determined in a postprocessing manner, by considering the values of $\Delta\lambda^\alpha$. Those greater than zero can be defined as active, but this only works if the problem is solved exactly. Through the Newton–Raphson procedure the problem is never solved exactly, so this strict requirement cannot be implemented. One could use a softer requirement by stating that slip systems with $\Delta\lambda^\alpha > tol$ are considered active. However, this only raises more questions about how this tolerance should be determined. This tolerance should depend on the level of convergence of the Newton–Raphson method, which itself is not easly to determine, and more questions arise about the details of the dependence. Therefore, because this treatment

[6]In this section when we describe the initial guess as being "fine", we mean that there will be no discrepancy between the result for $\Delta\lambda^\alpha$ for this method, and methods using the initial guess $\Delta\lambda^\alpha = 0$

lacks a practical usable definition of the active set, we choose not to modify this algorithm.

Algorithm 7: Modified augmented Lagrangian algorithm (MALA)

Data: ε_n^p, A_n, χ_n^α, $\Delta\lambda_n^\alpha$, $\mathcal{A}_n$
Result: ε^p, $\Delta\lambda^\alpha$
if $\mathcal{A}_n \neq \emptyset$ *and* $\max M^\alpha\left(\Delta\lambda^\beta = \Delta\lambda_n^\beta\right) < \max M^\alpha\left(\Delta\lambda^\beta = 0\right)$ *and* ($\phi^\alpha(\Delta\lambda_n^\alpha) > 0$ *when* $\Delta\lambda_n^\alpha > 0\ \forall\alpha$) **then**
  $\Delta\lambda_{(1)}^\alpha = \Delta\lambda_n^\alpha$, $\eta_{(1)} = \eta_0$, $i = 1$, $j = 1$
end
else
  $\Delta\lambda_{(1)}^\alpha = 0$, $\eta_{(1)} = \eta_0$, $i = 1$, $j = 0$
end
while $\max\phi^\alpha < tol$ **do**
  (†) **while** *First iteration or* $\max M^\alpha < tol$ **do**
    Construct residual:
    $M^\alpha\left(\Delta\lambda_{(i+1)}^\beta\right) = \Delta\lambda_{(i+1)}^\alpha - \max\left[0, \Delta\lambda_{(i)}^\alpha + \Delta t\eta_{(i)}\phi^\alpha(\Delta\lambda_{(i+1)}^\beta)\right]$
    Construct Jacobian:
    $$K^{\alpha\beta} = \delta_{\alpha\beta} - \begin{cases} 0 & \text{if} \quad \max\left[0, \Delta\lambda_{(i)}^\alpha + \Delta t\eta_{(i)}\phi^\alpha\right] = 0 \\ \Delta t\eta_{(i)}\frac{\partial\phi^\alpha}{\partial\Delta\lambda_{(i+1)}^\beta} & \text{otherwise} \end{cases}$$
    Update: $\Delta\lambda_{(i+1)}^\alpha = \Delta\lambda_{(i+1)}^\alpha - \left(K^{\alpha\beta}\right)^{-1} M^\alpha$
    $A = A_n + \sum_\alpha \Delta\lambda_{(i+1)}^\alpha$
    $\chi^\alpha = \chi_n^\alpha + \sum_\beta h^{\alpha\beta}(A)\Delta\lambda^\beta$
  end
  if $j == 1$ *and* $\mathcal{A}_n \neq \emptyset$ **then**
    $\mathcal{A}_{(i+1)} = \left\{\alpha \in \mathcal{P} | \max\left[0, \Delta\lambda_{(i+1)}^\alpha + \Delta t\eta\phi^\alpha(\Delta\lambda_{(i+1)}^\alpha)\right] \neq 0\right\}$
    if $\mathcal{A}_{(i+1)} \not\subset \mathcal{A}_n$ **then**
      $\Delta\lambda_{(1)}^\alpha = 0$, $\eta_{(1)} = \eta_0$, $i = 1$, $j = 0$
      go to (†)
    end
  end
  $\eta_{(i+1)} = m\eta_{(i)}$, $i = i + 1$
end

6.4 Algorithmic elastoplastic moduli

An important ingredient for an efficient finite element implementation is the algorithmic elastoplastic moduli. This was illustrated in (5.24). Here we are interested in calculating

$$\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}} = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}^{e*}}$$

where $\boldsymbol{\varepsilon}^{e*}$ is the elastic trial strain. We use the stress relationship

$$\boldsymbol{\sigma} = \mathcal{C}^e \boldsymbol{\varepsilon}^{e*} - \sum_{\alpha \in \mathcal{A}} \Delta\lambda^\alpha \left(\mathcal{C}^e \boldsymbol{P}^\alpha\right).$$

Taking the derivative of this expression with respect to $\boldsymbol{\varepsilon}^{e*}$,

$$\mathcal{C}^{ep} = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}^{e*}} = \mathcal{C}^e - \sum_{\alpha \in \mathcal{A}} \left(\mathcal{C}^e \boldsymbol{P}^\alpha\right) \otimes \frac{\partial \Delta\lambda^\alpha}{\partial \boldsymbol{\varepsilon}^{e*}}.$$

We now require an expression for $\frac{\partial \Delta\lambda^\alpha}{\partial \boldsymbol{\varepsilon}^{e*}}$. At this point the numerical method by which we calculate $\Delta\lambda^\alpha$ becomes important.

6.4.1 Rate-independent approach: SSA, EA, ALA, and MALA

All rate independent approaches enforce $\phi^\alpha(\Delta\lambda^\alpha) = 0 \; \forall \; \alpha \in \mathcal{A}$ exactly. We use this to find an expression for $\frac{\partial \Delta\lambda^\alpha}{\partial \boldsymbol{\varepsilon}^{e*}}$ by using the chain rule

$$\frac{\partial \Delta\lambda^\alpha}{\partial \boldsymbol{\varepsilon}^{e*}} = \sum_{\beta \in \mathcal{A}} \frac{\partial \Delta\lambda^\alpha}{\partial \phi^\beta} \frac{\partial \phi^\beta}{\partial \boldsymbol{\varepsilon}^{e*}}.$$

The term $\frac{\partial \Delta\lambda^\alpha}{\partial \phi^\beta}$ is just the inverse of (6.9). The term $\frac{\partial \phi^\beta}{\partial \boldsymbol{\varepsilon}^{e*}}$ is determined using (6.6), that is

$$\frac{\partial \phi^\beta}{\partial \boldsymbol{\varepsilon}^{e*}} = (\boldsymbol{P}^\beta : \mathcal{C}^e).$$

So the expression for the algorithmic elastoplastic moduli is given by

$$\mathcal{C}^{ep} = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}^{e*}} = \mathcal{C}^e - \sum_{\alpha \in \mathcal{A}} \sum_{\beta \in \mathcal{A}} (K^{\beta\alpha})^{-1} (\mathcal{C}^e \boldsymbol{P}^\alpha) \otimes (\boldsymbol{P}^\beta \mathcal{C}^e) \tag{6.25}$$

6.4.2 Rate-dependent approach: VA

In this formulation we found $\Delta\lambda^\alpha$ that satisfy (6.16). Once again we use this to find an expression for $\frac{\partial \Delta\lambda^\alpha}{\partial \boldsymbol{\varepsilon}^{e*}}$ by using the chain rule

$$\frac{\partial \Delta\lambda^\alpha}{\partial \boldsymbol{\varepsilon}^{e*}} = \sum_{\beta\in\mathcal{A}} \frac{\partial \Delta\lambda^\alpha}{\partial M^\alpha}\frac{\partial M^\alpha}{\partial \boldsymbol{\varepsilon}^{e*}}.$$

The term $\frac{\partial \Delta\lambda^\alpha}{\partial M^\alpha}$ is just the inverse of (6.17). The term $\frac{\partial M^\alpha}{\partial \boldsymbol{\varepsilon}^{e*}}$ is determined simply by differentiation of (6.16):

$$\frac{\partial M^\alpha}{\partial \boldsymbol{\varepsilon}^{e*}} = (\boldsymbol{P}^\beta : \mathcal{C}^e).$$

So the expression for the algorithmic elastoplastic moduli is given by

$$\mathcal{C}^{ep} = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}^{e*}} = \mathcal{C}^e - \sum_{\alpha\in\mathcal{A}}\sum_{\beta\in\mathcal{A}} (K^{\beta\alpha})^{-1}(\mathcal{C}^e\boldsymbol{P}^\alpha)\otimes(\boldsymbol{P}^\beta : \mathcal{C}^e) \tag{6.26}$$

Chapter 7

Implementation in an object orientated framework

In recent years there has been interest in developing finite element software using an object-orientated framework. There has been much developement in this with several open source options being developed, for instance fem++, fenics, deal.II, to name a few [2]. There has also been some intrest in the development object-orientated programs for plasticity, see for example [45, 16, 15, 13]. In this chapter we will driefly discuss the object-orientated paradigm, the open souce finite element library deal.II, and suggest a program layout for the problem of crystal plasticity.

7.1 What is object-orientated programming?

The two major programming paradigms that are used today are the procedural and object-orientated paradigms. They are loosely defined as:

- **Procedural:** The problem is decomposed into individual procedures or subroutines. This decomposition is usually done in a top-down manner. In a top-down approach, once a section of the problem has been identified as being implementable by a procedure, it too is broken down into individual procedures. The data however, is not usually part of this decomposition.

- **Object-orientated:** The problem is decomposed into interacting objects. Each object encapsulates and hides methods that manipulate the hidden state of the object. A message sent to an object invokes the encapsulated method that then performs the requested task.

The object-orientated paradigm was developed as a result of an increase in computational capacity of hardware. Programmers could now construct significantly larger programs. In an attempt to organise these large programs the object-orientated paradigm was developed. The key feature of this paradigm has been that of a class, an instance of which is called an object. An object is seen as a pseudo-self sufficient entity, which contains all data and procedures to perform its specific task. This emphasis on keeping data within the object is what contrasts it from a procedural language, where this is not done. This kind of data is referred to as private. One interacts with an object through its public methods, which operate on the private data. A well structured object *encapsulates* its data.

One can define relationships between classes depending on the object-orientated languages used. Typical relationships found in most object-orientated languages are inheritance and polymorphism. An inherited relationship is when one class inherits all the members (data and methods) from another class. It is often seen as an "is a" relationship. Typical examples would be "a bull-mastiff is a dog". The class *Dog* would set out all the attributes every dog should have, things like bark or run. The class *Bull-mastiff* would inherit these attributes either directly or redefine some of them. The *Bull-mastiff* class would also be free to add additional attributes that not all dogs have, for example drool. Loosely put polymorphism is getting an object of a certain type to behave as an instance of a derived class. A typical object-orientated program consists of multiple objects interacting with one another through their public interfaces.

One of the biggest advantages of the object-orientated paradigm is that it leads itself to creating routines that can be easily be re-used. It is considered by many to be the standard paradigm for implementing large scale projects, though it does have its critics. Many of the criticisms are focused on the programmer having to create overly complicated data structures as a result of the restrictions the paradigm imposes on data: "The problem with object-oriented languages is they've got all this implicit environment that they carry around with them. You wanted a banana but what you got was a gorilla holding the banana and the entire jungle." [38]

In most popular object-orientated languages one can find a large amount of 3rd party software. This 3rd party software is often in the form of a library, consisting of many classes.

For our purposes, we are interested in a library that will perform the finite element analysis. For our purposes the open-source finite element library deal.II [7], which is implemented in C++, is used. A brief introduction to deal.II that highlights the relevant aspects is given in the following section.

7.2 Open-source finite element library deal.II

Deal.II (Differential Equations Analysis Library) is a tool for solving differential equations using finite element method. It is a very large library written in C++ that is still actively being developed. In this section we will discuss the program flow and class collaboration that is required for a problem to be solved using finite elements in deal.II. We discus the concept behind each class or step in order to understand the role that it plays in the final solution. A general overview of the program flow, relationships between classes and the dependence of these classes is shown in Figure 7.1.

We start our discussion with the *Unit cell*. We referred to this as the reference element in Section 5.6. In deal.II the only possible reference element is the unit hypercube $[0,1]^n$. For problems in mechanics this would typically be the unit line $[0,1]$ in 1D, the unit square $[0,1]^2$ in 2D and the unit cube $[0,1]^3$ in 3D. Deal.II has no support for other reference elements such as triangles, tetrahedra, pyramids or prisms. This restricts the user when partitioning the domain, as described in section 5.4, since now the partitioned domain must consist of elements that can be mapped to a hypercube. The desired dimension of the problem, which defines the hypercube dimension, is stipulated by the user. The user can access various geometric properties about the *Unit cell* though the *GeometryInfo* class, which contains information about the number of vertices per cell, the ordering of faces and the direction of edges to name a few.

Next we discuss the *Triangulation* class. Essentially this class contains information about the domain, or rather the partitioned domain, over which the problem is stated. For example if the problem was to find the resulting displacement of a copper bar with length 0.3m and

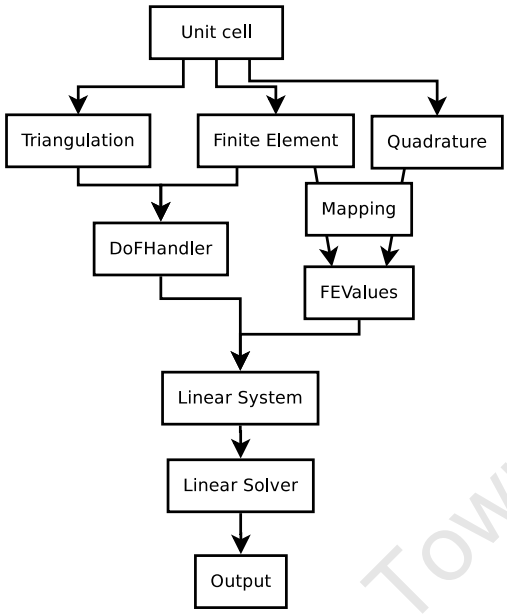


Figure 7.1: Collaboration diagram of the most important steps and classes in a deal.II finite element implementation [1].

radius 0.01m for some applied load, then the triangulation class would contain the geometric information of the partitioned domain, being the mesh of the bar. This class would not have any information on the material type, applied loads or the shape function that is used. It knows the position and the connectivity of the vertices for the mesh. Once the triangulation class is set up, the user will often use it to iterate over cells or faces. For instance if one wished to construct $\boldsymbol{F}^{\text{ext}}$ in (5.20c) then one iterates over the N elements and construct $\boldsymbol{h}^{(e)}$. Iterating over the elements is done by *cell iterators*; a pointer like object, supplied by the *Triangulation* class.

The *Finite element* classes contain all the properties of the desired shape function. This class is defined relative to the dimension of the problem or the *Unit cell*. The *Finite element* classes know nothing about the domain of the problem, the boundary conditions, etc. It specifies the position of nodes on the reference element, the form of the shape function $\boldsymbol{\psi}_i(\boldsymbol{x})$ and gradient of the shape function specified relative to the reference element. The user seldom uses the *Finite element* classes apart from declaring the desired shape function. It can be used to access information about the number of degrees of freedom associated with each cell, node or line. It can also provide the user with information about the value and

gradient of individual shape functions at any point within the element.

The *Quadrature* class contains all information necessary to perform the numerical quadrature on the reference element. This is information such as the quadrature point positions on the reference element and its associated weight.

The *DoFHandler* brings together information provided by the *Triangulation* class and aspects of the *Finite element* class. From the *Triangulation* class information is utilised about the domain or mesh; in fact the *DoFHandler* class derives from *Triangulation* class so it inherits all its data and methods. From the *Finite element* class information is utilised about the chosen shape function; such as the position of nodes on the reference element and number of nodes per element. It uses this to distribute the nodes over the partitioned domain or mesh and gives them a global numbering. This class does not know anything about how the elements relate to the reference element or anything about the shape functions themselves. Interaction with this class is done in a similar way to the *Triangulation* class. The *DoFHandler* class provides a cell iterator through which one accesses information about each cell, face or line, etc. One can access all information of a topological and geometric nature of the partitioned domain, in addition to things like the global numbers of the degrees of freedom on the present cell.

The object that handles the transfer of information from the reference element to an element is the *Mapping* class and its derived classes. The two classes that are defined with respect to the reference element; the *Quadrature* and *Finite Element* classes utilise the *Mapping* class to map points to and from the reference element and the elements. For instance, if one wishes to find the value of a shape function at a point on the mesh, then that point is mapped to the reference element where the shape function value is determined. The *Mapping* classes also provided information about the Jacobian for a particular mapping.

In the implementation of finite elements, one is often interested in integrating the basis function over an elements domain, for example (5.20e). As we eluded to in (5.25) we use Gauss quadrature to evaluate these integrals. The *FEValues* class provides the functionality to achieve this by bringing together aspects of the *Finite Element* and *Quadrature* classes. It provides the value of the shape function and shape function gradients at quadrature points, as well as Jacobian and quadrature weights. The *Finite Element* and *Quadrature* classes are defined on the reference element, and we need these values on the actual elements, so

the *FEValues* class uses the *Mapping* class to achieve this.

With the tools provided above one can construct the linear system described by (5.23). The *DoFHandler* class allows us to iterate over cells and the degrees of freedom. The *FEValues* class facilitates the calculation of the elements contribution to the global system matrices and vectors. Then the *DoFHandler* knows where the elements contributions belong in the system matrices and vectors.

Once the entire linear system is constructed the user then needs to solve for the systems unknowns. This often requires the use of a *Linear Solver*. Deal.II provides a great number of direct and iterative solvers. There are also a great number of third party solvers that deal.II can utilise.

Once a solution has been found one may want to post process it, using visualisation software. Deal.II is capable of outputting data in many output formats, which can be viewed using third party visualisation software such as Paraview [3] (an open source scientific visualization tool).

One can find extensive documentation of this library at `www.dealii.org`.

7.3 Program layout and data structures

Now that we have discussed the implementation of a finite element code we turn towards the implementation of the crystal plasticity formulation. As we have started in Section 5.6 it is only necessary to know the stress at the Gauss points. The algorithms we have developed in section 6 are valid at any point. All that these algorithms need to know is a limited set of information at a point, the result of which is a stress at that point; for this reason these algorithms are referred to as local algorithms. Here we shall discuss the data structures we used for there implementation.

The first class we shall discuss is the *SlipSystem* class. This class' main purpose is to manage all data that is relevant to a slip system. The relevant data differs slightly from formulation to formulation. For instance, all algorithms have data λ^α, $\Delta\lambda^\alpha$, χ_n^α and $\boldsymbol{P}^\alpha$. The augmented Lagrangian algorithm requires an additional data member $\Delta\lambda_{(i)}^\alpha$ and minor additional functionality. It makes sense that we have a base class *SlipSystem* that most

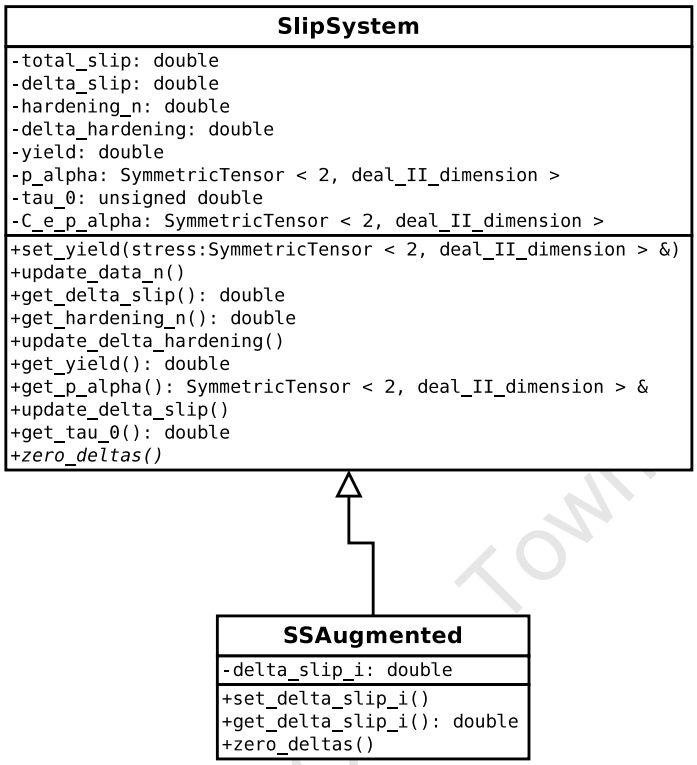


Figure 7.2: Proposed data structure for the *SlipSystem* class.

of the algorithms use, and a derived class *SSAugmented* that the augmented Lagrangian algorithm uses. The data structure is illustrated in Figure 7.2.

The next class is the *SetHandler* classes. The purpose of these classes is to manage the *SlipSystem* objects; in a similar manner as the *Triangulation* and *DoFHandler* manages cells. The main member is a collection of the *SlipSystem* objects, in this case a `std::vector` of *SlipSystem* objects. The different algorithms have different requirements on this class, but much of the functionality is similar. A central idea in the the penalty algorithm and the set search algorithm is the active set search. For this reason, we would like to have an additional data type that contains indices of the set of all *SlipSystem* objects that are considered active at that moment. This introduces additional methods to interact with the active set. For this reason, we derive a class *SHSetSearch* from *SetHandler* to add the additional functionality. As a result of the augmented algorithm using the *SSAugmented* class, which has more functionality, we need a derived class *SHAugmented* of *SetHandler* to utilise these functions. The user can iterate over the *SlipSystem* objects and interact with

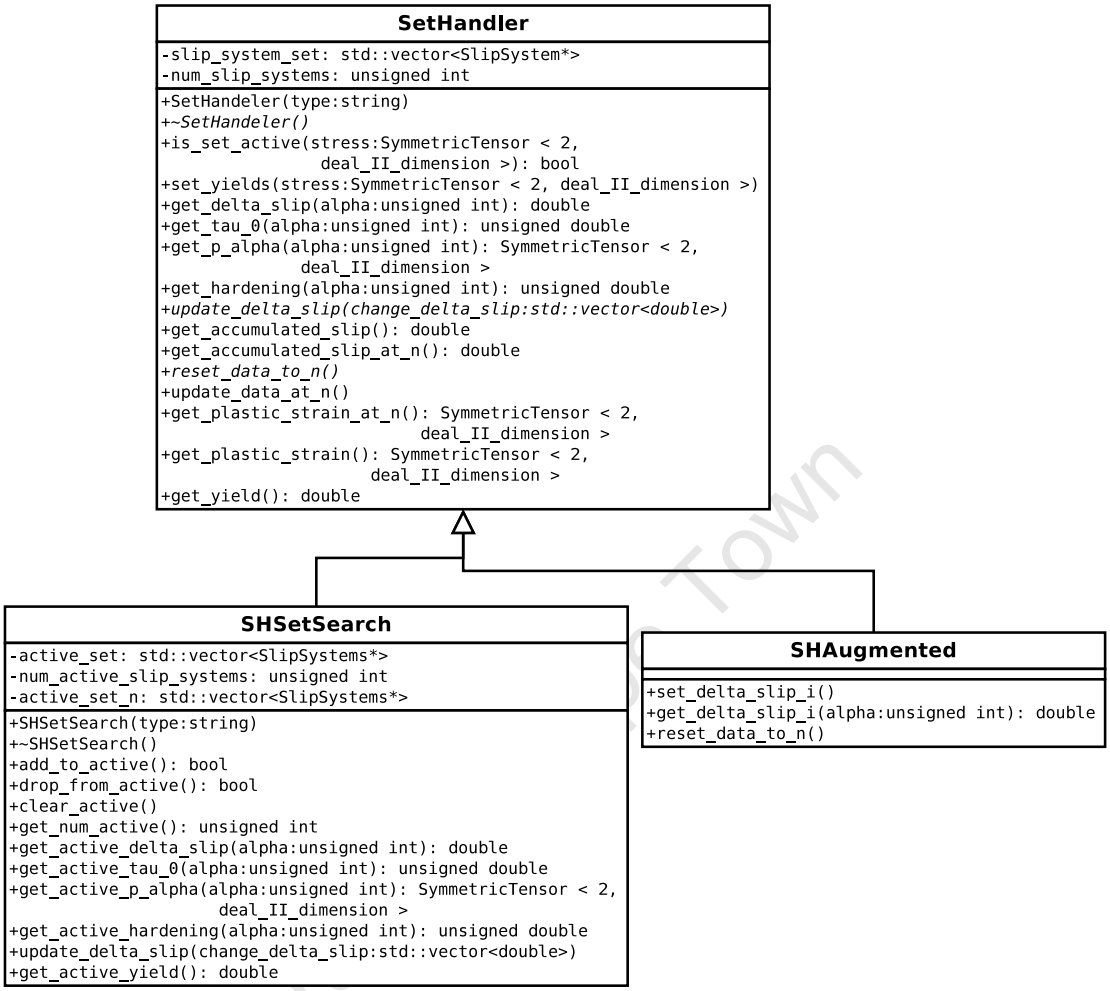


Figure 7.3: Proposed data structure for the *SetHandler* class.

a *SlipSystem* through its methods. Where possible the *SetHandler* can perform tasks on an entire set. The data structure is illustrated in Figure 7.3.

The *HardeningModel* classes are responsible for providing information about the hardening model. The class should provide information about the hardening moduli $h^{\alpha\beta}$ and as well as the update to the hardening term in (6.2). This functionality is provided by the base class. The derived classes define the hardening type, i.e. provide a specific form for $h(A)$. The data structure is illustrated in Figure 7.4. Here we provided, as an example, the derived classes for the hardening laws described in (3.17). If required, additional hardening laws could easily be implemented by deriving additional classes from the base class *HardeningModel*.

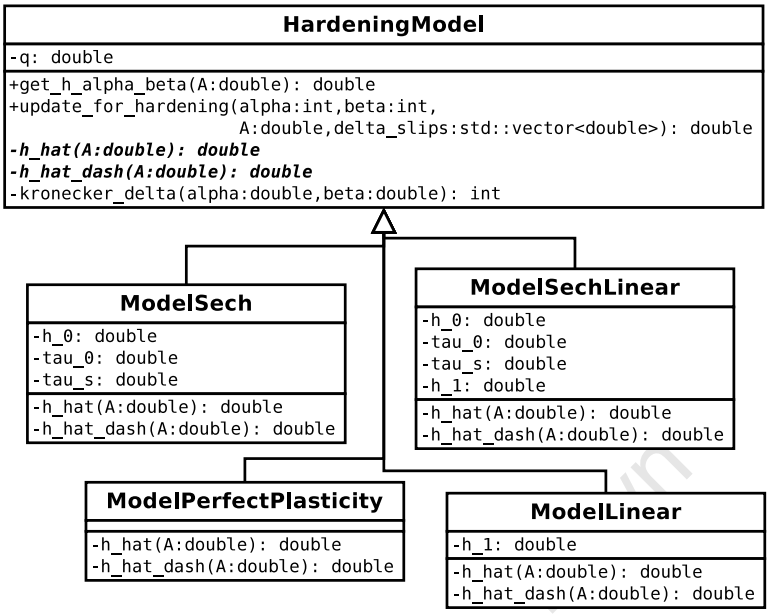


Figure 7.4: Proposed data structure for the *HardeningModel* class.

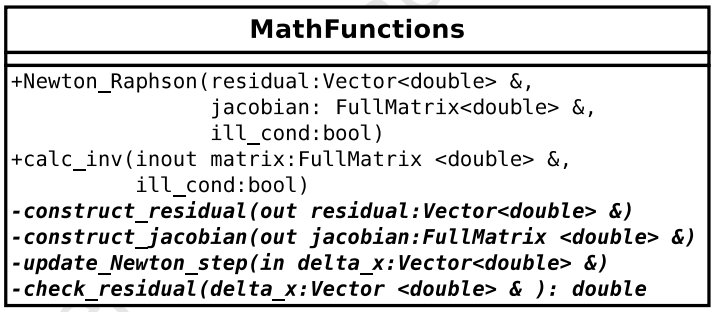


Figure 7.5: Proposed data structure for the *MathFunctions* class.

All the algorithms that have been discussed utilise a Newton–Raphson method, and require the inversion of a matrix to construct the algorithmic elastoplastic moduli (6.25)-(6.26). The class that provides this functionality is the *MathFunctions* class. The numerical methods discussed in Section 4 are implemented in the *MathFunctions* class. The data structure is illustrated in Figure 7.5. The `Newton_Raphson` method, performs a globally convergent algorithm described in Section 4.3.1. The solver utilised in the Newton–Raphson scheme is notified that the system is potentially ill-conditioned or singular though the boolean `ill_cond`. In the case of an ill-conditioned system, the Newton–Raphson uses an SVD solver described in Section 4.2. If one wishes to use the Newton–Raphson capabili-

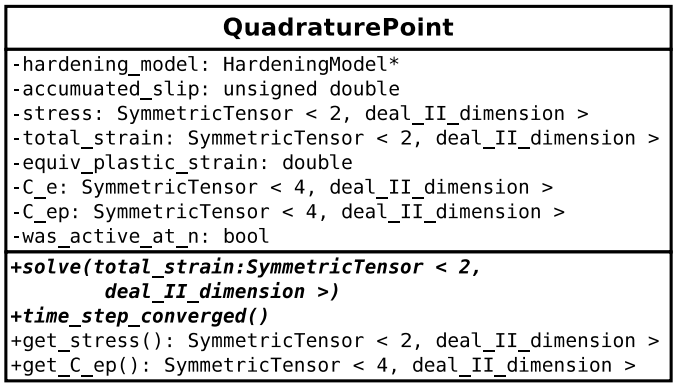


Figure 7.6: Proposed data structure for the *QuadraturePoint* base class.

ties of this class, one must derive a class that defines the functions `construct_residual`, `construct_jacobian`, `update_Newton_step` and `check_residual`.

The final classes to be discussed here are the *QuadraturePoint* classes. These classes implement the desired algorithm. They contain a *HardeningModel* and a *SetHandler* and utilise these in the implementation. The base data structure is illustrated in Figure 7.6. Interaction with these classes is though the methods:

- `solve` which, for a given trial strain, calculates the stress by a predetermined algorithm.
- `time_step_converged` sends a message to the *SlipSystem* that a new time step has converged so that it can store it's data appropriately.
- `get_stress` returns the stress at the quadrature point.
- `get_C_ep` returns the algorithmic elastoplastic moduli.

A data structure that finds the stress by the algorithms given in Section 6.3 is given in Figure 7.7.

One may ask the question how does this data structure interact with the data structure available in deal.II? We use the void user pointer available to every cell in the *DoFHandler* class. In short, sets of *QuadraturePoint* objects (having a size equal to the number of quadrature points in that cell) are allocated to a cell's user pointer. Each quadrature point is then automatically associated with a *QuadraturePoint* object.

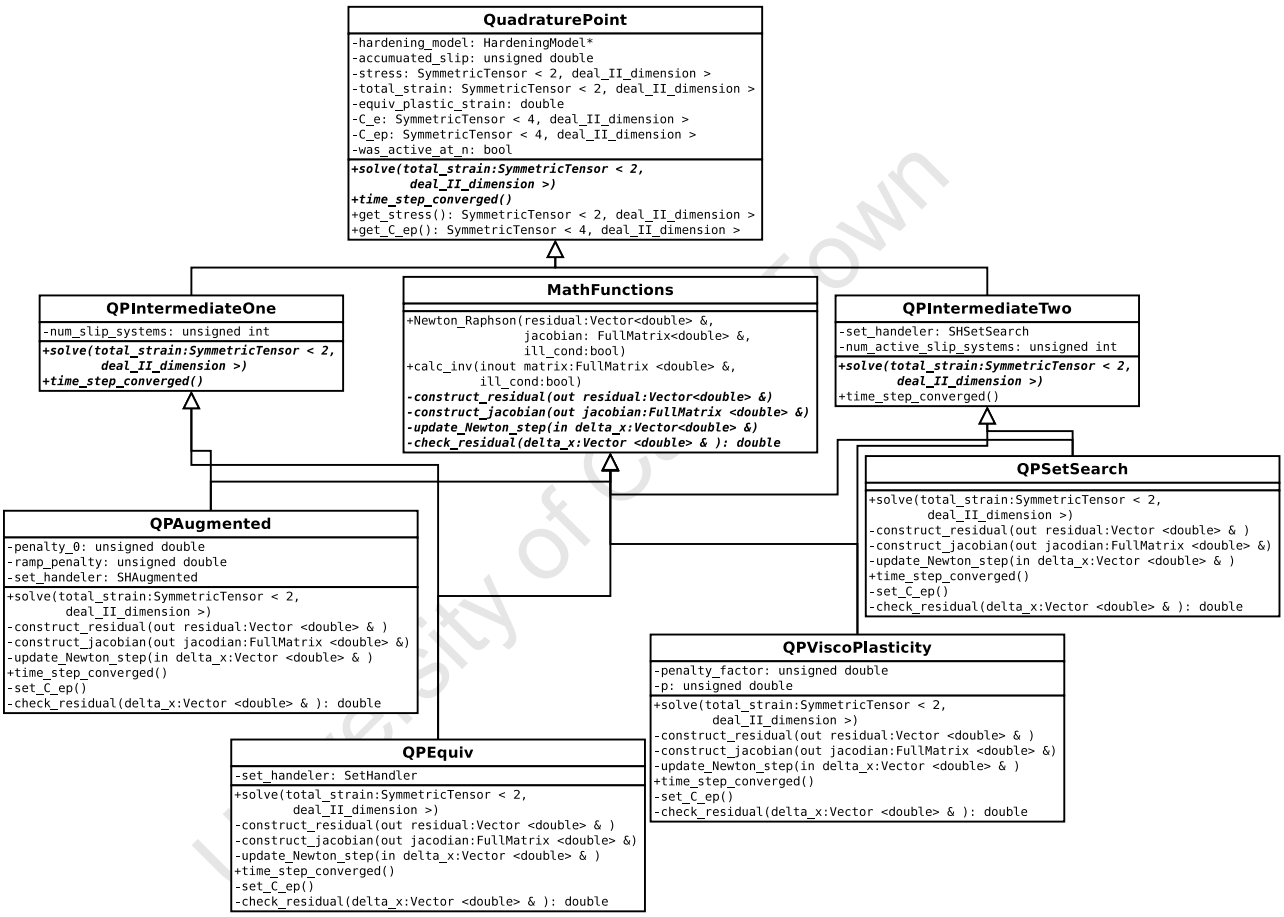


Figure 7.7: Proposed data structure for the *MathFunctions* class.

Part III

Numerical results and conclusions

Chapter 8

Numerical examples

In this chapter, we present results based on the algorithms presented in Chapter 6. We present two experiments. The first is a small-scale problem of the shearing of a cube. This is an idealised example, but still significantly challenging. It is intentionally implemented in such a way that there is uniform stress throughout the cube, thus the results are easily analysed at the quadrature point level. The main objective of this numerical experiment is to compare and contrast the quadrature point results of the various algorithms. We also compare the results to those found in the literature. The second is a computationally larger and more complex problem of a tension test. An intentional weakness is introduced into the strip such that a strongly localised bands of slip are induced. Here there are several objectives. The first is to compare these results with those found in the literature. The second is to compare the agreement between the algorithms. For this purpose, several results are prepared. Lastly, we compare the computational efficiency of the various algorithms. This is done to establish, in a quantitative manner, which of the rate-independent algorithms is computationally superior. In addition, we establish if the modification to the ALA is effective.

8.1 Cube in shear

A perfectly plastic cube composed of an f.c.c. material is subjected to perfect shear. The boundary conditions are chosen to replicate a state of plane strain. This problem has been

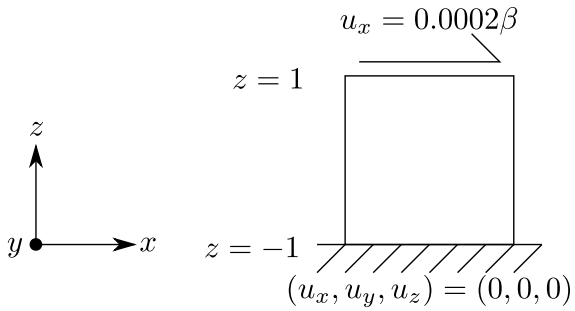
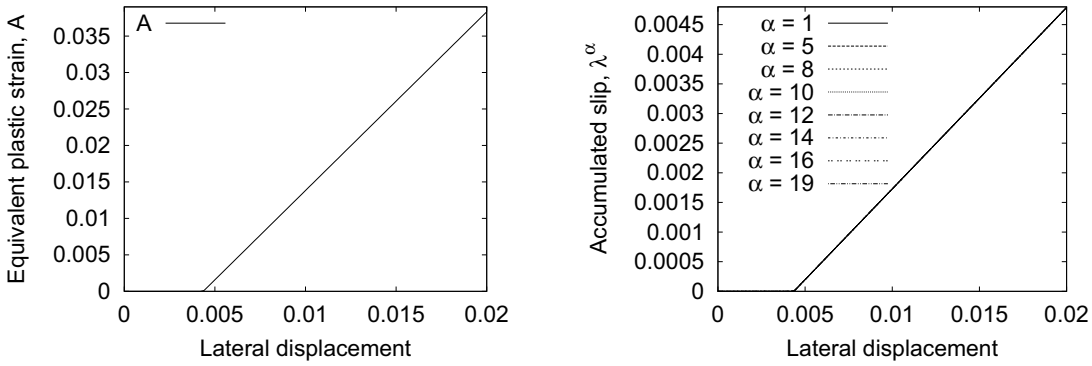


Figure 8.1: The displacement boundary conditions for the shearing of a cube.

Hardening model	
perfect plasticity	$h = 0$
critical resolved shear stress	$\tau_0^\alpha = \tau_0 = 1.0$ N/mm^2
Elastic tensor	isotropic
bulk modulus	$\kappa = 1500.0$ N/mm^2
shear modulus	$\mu = 562.5$ N/mm^2
VA	
strain-rate sensitivity	$p = 200$
penalty (viscosity)	$\eta = 2000$
ALA/MALA	
initial penalty	$\eta_{(1)} = 0.5$

Table 8.1: Shearing of a cube model parameters.

investigated in [29] using the SSA and VA . For simplicity, we assume an elastically isotropic crystal with a bulk modulus $\kappa = 1500.0$ N/mm^2 and shear modulus $\mu = 562.5$ N/mm^2. The initial resistance to plastic flow is chosen to be the same for all slip systems $\tau_0^\alpha = \tau_0 = 1.0$ N/mm^2. In a deformation-driven process the cube's top surface ($z = 1$) is deformed in 100 equal time steps in the x-direction. The displacement of the top surface relative to the initial configuration is give by $u = 0.0002\beta$, where β is the load parameter. The bottom surface ($z = -1$) is fixed. The boundary conditions are given in Figure 8.1. This test is performed for several orientations of the crystal structure. The initial orientation is chosen to be $\{\theta_1, \theta_2, \theta_3\} = \{0^\circ, 0^\circ, 0^\circ\}$. This configuration is then rotated about the z- and y-axes, in increments of 18°. For each orientation the problem is solved, and the final shear stress state and slip history recorded. This entire set of experiments is solved using the SSA, VA, EA, ALA and MALA approaches. A list of parameters can be found in Table 8.1. The choice for η in the VA is motivated by choosing a large enough penalty that a near



(a) (b)

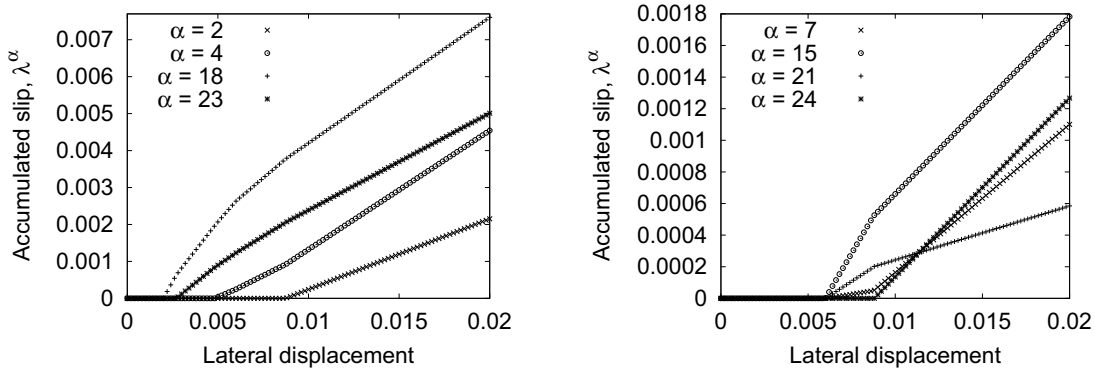
Figure 8.2: Results for crystal orientation $\{\theta_1, \theta_2, \theta_3\} = \{0°, 0°, 0°\}$. (a) The equivalent plastic strain versus total lateral displacement, (b) the accumulated slip versus total lateral displacement for slip systems with non-zero accumulated slip. These were produced using the ALA. The final stress was recorded to be 2.44949. Near-identical results were attained for all other algorithms (i.e. the SSA, EA, VA and MALA).

rate-independent solution is obtained, but not so large that the problem becomes singular.

For all the local algorithms, the final shear stress state and the history of plastic slip agrees with those presented in [29]. An example of the slip history is shown in Figure 8.2 and 8.3.

Figure 8.2(b) shows a set of eight slip systems experiencing the same amount of slip at each time step. This can be expected due to the orientation of the crystal structure and the nature of the total strain $\boldsymbol{\varepsilon}$ state. The initial total strain state (while still in the elastic regime) produces a stress that only has non-zero components at $\sigma_{12} = \sigma_{21}$, bearing in mind that the stress is symmetric. We are interested in the decomposition of this stress onto the slip systems, i.e.$\tau^\alpha = \boldsymbol{\sigma} : \boldsymbol{P}^\alpha$, as this is an indication of which slip systems will experience slip first, see (3.18). Initially, at least, we are interested in which α have the largest P_{12}^α component. These will have the greatest Schmidt stress, and will experience plastic deformation first. For this orientation, $P_{12}^\alpha = 0 \; \forall \; \{\alpha | \alpha = 3,\ 6,\ 9,\ 11,\ 15,\ 18,\ 21,\ 23\}$, $P_{12}^\alpha = -0.20412 \; \forall \; \{\alpha | \alpha = 2,\ 4,\ 7,\ 13,\ 17,\ 20,\ 22,\ 24\}$ and $P_{12}^\alpha = 0.20412 \; \forall \; \alpha \in \mathcal{H} = \{\alpha | \alpha = 1,\ 5,\ 8,\ 10,\ 12,\ 14,\ 16,\ 19\}$ [1]. Thus the set $\mathcal{H}$ will experience plastic slip first under these loading conditions. Consider the first time step where one needs to calculate

[1]Here we have numbered the slip systems 1-12 according to Table 3.1, numbers 13-24 are the doubly defined pairs of Table 3.1, i.e. slip system 1 has the doubly defined pair 13 and so on.



(a) (b)

Figure 8.3: The accumulated slip versus total lateral displacement for the ALA with crystal orientation $\{\theta_1, \theta_2, \theta_3\} = \{-18°, -54°, 0°\}$. Here we only only show the slip systems with non-zero accumulated slip. The data has been split into Figure 8.3(a) and 8.3(b) due to a scale change in the data. The final stress was recorded to be 1.46913. Near-identical were attained for all other algorithms (i.e. the SSA, EA, VA and MALA).

the slip for $\mathcal{H}$. The problem will be exactly the same on each slip system in $\mathcal{H}$, so we expect the solution to be the same on each slip system; let that be $\Delta\lambda_*$.

The next consideration is how the stress will be affected by the plastic deformation. In general a slip system which experiences slip in the current time step could have a considerable influence on the stress in future time steps. The matrix $\boldsymbol{P}^\alpha$ form a basis for $\Delta\boldsymbol{\varepsilon}^p$ (see for example (6.3a)). The scalars that form the linear combinations of these basis tensors are the $\Delta\lambda^\alpha$, so a change in $\Delta\lambda^\alpha$ causes a change in the composition of $\Delta\boldsymbol{\varepsilon}^p$ (and hence a change in $\boldsymbol{\varepsilon}^p$). For instance a considerable change in $\boldsymbol{\varepsilon}^p$ can occur when a slip system becomes active for the first time, i.e. a λ^α goes from being zero to a positive scalar. A change of the plastic strain ultimately effect the stress due to (3.15). The change that this can cause in the stress can have a drastic effect on other slip system's yield criterion values, allowing them to experience slip in future time steps. Thus a slip system experiencing plastic slip can encourage slip systems to experience slip in a future time step. The plastic strain for the first plastic iteration will be given by

$$\boldsymbol{\varepsilon}^p = \sum_{\alpha\in\mathcal{H}} \Delta\lambda_* \boldsymbol{P}^\alpha,$$

for this problem this would be

$$= \Delta\lambda_* \begin{pmatrix} 0 & 1.63299316 & 0 \\ 1.63299316 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Since $\boldsymbol{\sigma} = \mathcal{C}^e(\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^p)$, we can conclude that the non-zero components of the stress will remain so in the following time steps. So for future time steps we expect the same slip systems to experience plastic slip. This in some way explains the result in Figure 8.2.

The result in Figure 8.3 shows the complex and coupled behaviour of the slip history for the crystal orientation $\{\theta_1, \theta_2, \theta_3\} = \{-54°, -18°, 0°\}$. Notice how slip systems become active at later stages of the experiment and how multiple slip systems can become active in one time step. This kind of behaviour can be difficult to capture, and serves as motivation for a robust algorithm.

Some interesting deductions can be made from the fact that we are using a perfectly plastic model and proportional loading conditions. Imagine a set of time steps where the active set remains unchanged. Due to the proportional loading conditions, we can expect that the initial violation of the yield function is identical for these time steps. In fact, one is solving exactly the same problem in these time steps, thus we expect the solution to be exactly the same. Therefore, $\Delta\lambda^\alpha$ will be exactly the same for these two time steps. This means the change in gradient of λ^α is zero, implying a straight line in this situation. When the active set changes between time steps, we can expect a considerable change in the solution for $\Delta\lambda^\alpha$. Thus a change in the gradient of λ^α is expected at these points. Since the change in active set is a discrete occurrence, one can expect the λ^α to consist of continuous piecewise straight lines. This is noticed in Figure 8.3. Changes in gradients of λ^α correspond to changes in the active set.

8.2 Strip in tension

We next look at an example where localisation of plastic slip occurs. This is an example of a strip of material in tension. The strip has dimensions 6mm × 15.4mm × 0.3mm. A partitioned domain is generated for the strip consisting of 384 ($12 \times 32 \times 1$) triquadratic

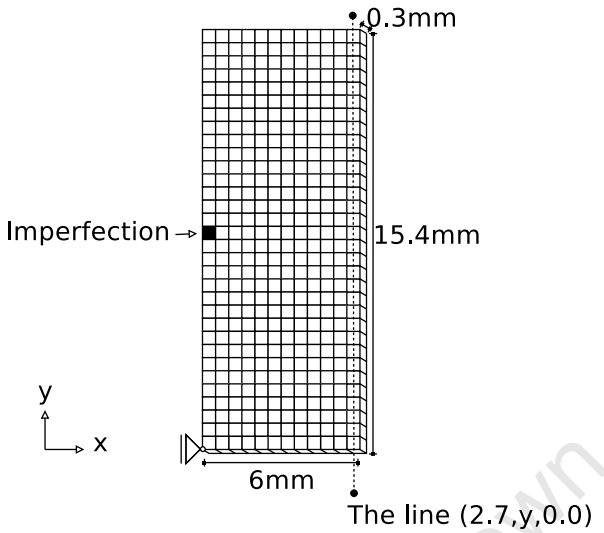


Figure 8.4: Schematic of a 6mm × 15.4mm × 0.3mm strip partitioned into 384 quadrilateral elements.

elements, see Figure 8.4. Here we used quadratic polynomial shape functions in order to circumvent the effect of locking associated with low-order trilinear elements.

Once again a displacement-driven process is used. A positive and negative displacement relative to the initial configuration of $u = \beta 15.4 \times 10^{-5}$mm are applied to the top and bottom surface respectively, where β is the load parameter. We stagger the elongation of the sample into nine equal load steps up to $\beta = 90$ followed by 110 equal load steps until a final load parameter of $\beta = 200$ is achieved. The initial steps are all elastic steps, where the history of the material is irrelevant. The displacement of the front and back walls are constrained to impose a state of plane strain. The side walls are allowed to contract in the x-direction. The x-displacement of the set of points in the bottom left corner are fixed. See Figure 8.5 for a visual representation of these boundary conditions.

We choose to model a f.c.c. crystal structure. The elastic response of the material is chosen to be anisotropic with cubic symmetries (see (3.22)). The hardening model is a sech+linear type, see (3.17), with a hardening parameter $q = 1.4$. The full set of material parameters is given in Table 8.2. Again, the choice for η in the VA is motivated by choosing a large enough penalty that a near rate-independent solution is obtained, but not so large that the problem becomes singular.

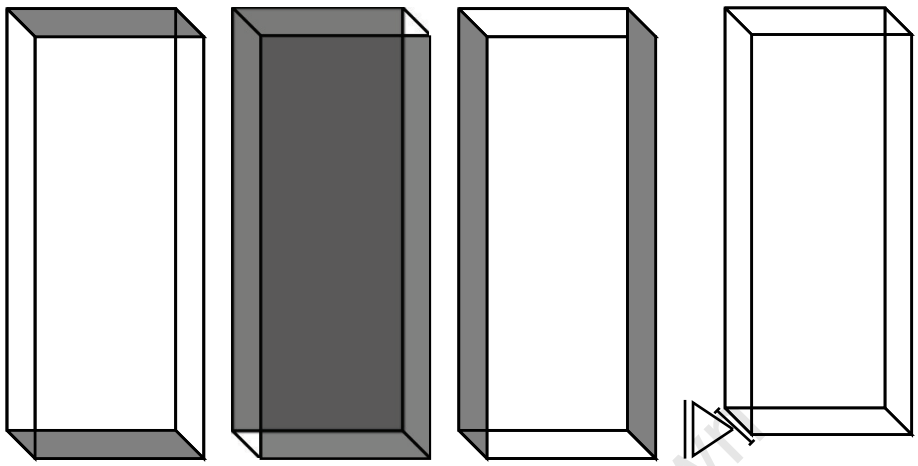
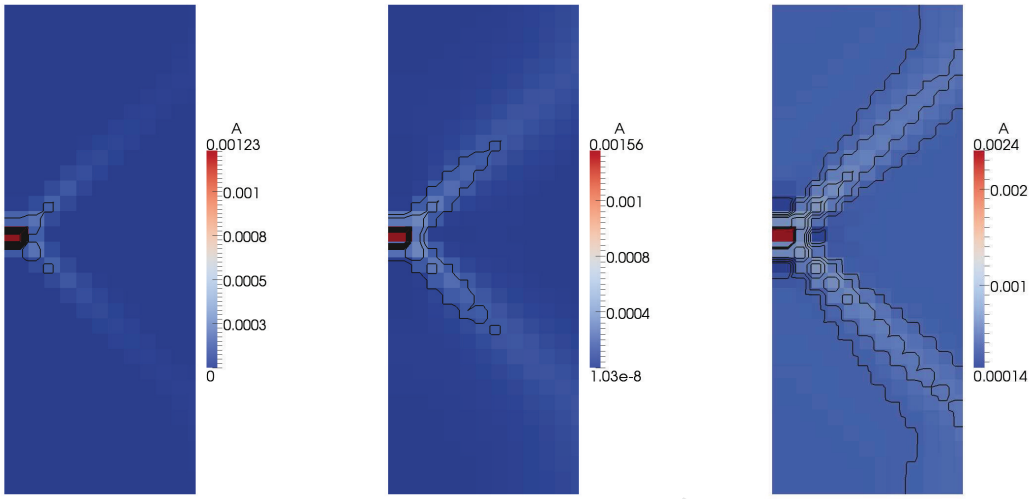


Figure 8.5: Loading and boundary conditions: (a) apply a positive and negative displacement of $u = \beta 15.4 \times 10^{-5}$mm in the y-direction to the top and bottom surfaces. (b) Fix the z-displacement on the front and back surfaces. (c) Left and right surfaces are free. (d) Fix the x-displacement of the points in the bottom left corner. Here, the orientation is the same as Figure 8.4.

Hardening model	
sech+linear	$h(A) = h_0 \cosh^{-2}\left(\frac{h_0 A}{\tau_\infty - \tau_0^\alpha}\right) + h_1$
initial hardening	$h_0 = 0.533$ GPa
linear hardening	$h_1 = 0.001$ GPa
critical resolved shear stress	$\tau_0^\alpha = \tau_0 = 0.060$ GPa
Saturation stress	$\tau_s = 0.108$ GPa
Hardening parameter	$q = 1.4$
Elastic tensor	anisotropic with cubic symmetries
moduli	$C_{11} = 107.300$ GPa
moduli	$C_{12} = 60.900$ GPa
moduli	$C_{44} = 28.300$ GPa
VA	
strain-rate-sensitivity	$p = 200$
penalty (viscosity)	$\eta = 2000$
ALA/MALA	
initial penalty	$\eta_{(1)} = 0.5$

Table 8.2: Tension test model parameters.



(a) (b) (c)

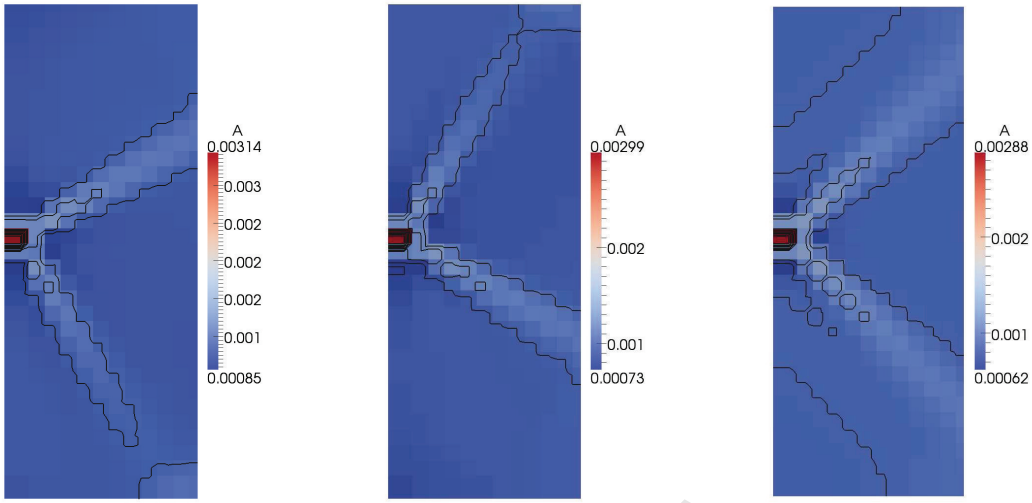
Figure 8.6: The equivalent plastic strain for the crystal orientation $\{\theta_1, \theta_2, \theta_3\} = \{0°, 0°, 0°\}$ at loading parameter (a) $\beta = 101$ (b) $\beta = 102$ (c) $\beta = 110$. Here, the orientation is the same as Figure 8.4.

In order to trigger the onset of localisation of the slip, a imperfection is placed in the material at the point illustrated in Figure 8.4. Within this cell we have reduced the yield stress by 10%.

8.2.1 Comparing the results for the various formulations to those found in literature

The first test that we run is to establish that the various formulations give the same results. The orientation of the crystal is chosen to be $\{\theta_1, \theta_2, \theta_3\} = \{0°, 0°, 0°\}$. The result displaying the distribution of the equivalent plastic strain $A = \sum_\alpha \lambda^\alpha$ over the domain is shown in Figure 8.6. This particular set of results were found using the EA. The other algorithms gave comparatively indistinguishable results. It can be seen in Figure 8.6 that there is a localisation of the equivalent plastic strain into two bands. These bands are called *slip bands*. The same phenomenon is present in the tension test simulated in [29].

In the next test we rotated the orientation of the crystal structure. Here, as in [29], we hope to observe the change in orientation of the slip bands. In Figure 8.7 we display results



(a) (b) (c)

Figure 8.7: The equivalent plastic strains for the loading parameter $\beta = 120$ and crystal orientation $\theta_2 = \theta_3 = 0°$ (a) $\theta_1 = 0°$ (b) $\theta_1 = -15°$ (c) $\theta_1 = 15°$. Here, the orientation is the same as Figure 8.4.

at the same loading parameter, but with different crystal orientations. Here we choose to display the crystal orientations $\{\theta_1, \theta_2, \theta_3\} = \{0°, 0°, 0°\}$, $\{15°, 0°, 0°\}$, $\{-15°, 0°, 0°\}$. In Figure 8.7 a strong correlation between the change in slip band angle and the change in orientation of the crystal structure is observed.

The numerical values given in a similar test conducted in [29], could not be reproduced. This could be due to two reasons. Firstly, we may have used a different hardening rule as details of the rule used in [29] are absent. Secondly, the interpolation used in the visualisation of results plays a large role in the displayed numerical values. If the interpolation of the results is different then the displayed numerical values could differ significantly. Despite this, the agreement of the results in Section 8.2 with those in [29] and the agreement in general behaviour (the development of slip bands, and the dependence of the slip band orientation on the crystal orientation) with those presented in [29], serve as motivation for confidence in our results.

8.2.2 A further comparison of the results of the various formulation

We wish to make a closer comparison of the various formulations by comparison with the visocoplastic solution. To do this, the cell averaged equivalent plastic strain has been plotted along a line. The position of this line must be such that it maximises the discrepancy in the solutions produced by the various formulations. To choose the position of this line we will conduct a small thought experiment. Consider a point P experiencing plastic deformation. The neighbouring point to P will be influenced by P because the plastic deformation of P has a nonlocal effect expressed though the stress field. The hardening of point P also plays a role in stimulating the neighbouring points to experience plastic deformation. So, plastic deformation should spread along the domain to neighbouring points. If there is an "error"[2] associated with calculating the slip then this "error" will be compounded as the plastic deformation spreads over the domain. In our case we know that the plastic deformation will start at the imperfection and then spread over the domain, thus we choose to plot the equivalent plastic strain over the line defined by $x = 2.7,\ z = 0.0$ (see Figure 8.4).

For simplicity we present and discuss results for crystal orientation $\{\theta_1, \theta_2, \theta_3\} = \{0°, 0°, 0°\}$. The result is shown in Figure 8.8. Figure 8.8(a) is chosen to be the first loading parameter for which these cells experience plastic slip. As one can see there is a significant discrepancy between the ALA/MALA solution and the other formulations at this load parameter. However, this discrepancy becomes small within a few time steps. By $\beta = 110$ the results are near identical (see Figure 8.8(f)). One interesting characteristic of the ALA/MALA solution is that it shows a much more pronounced localisation at earlier time steps. Similar behaviour was noted in crystal orientations $\{\theta_1, \theta_2, \theta_3\} = \{15°, 0°, 0°\},\ \{-15°, 0°, 0°\}$.

Another comparison that can be done is to compare the applied load versus displacement. This is only done for the top surface of the strip. The results can be seen in Figure 8.9. Here too we see a high degree of correlation between all the algorithms.

8.2.3 Efficiency test

The last test compares the computational times taken by the various rate-independent algorithms. We compare these times relative to the VA. This is done by dividing the

[2]By "error" we mean a discrepancy relative to the visocoplastic solution.

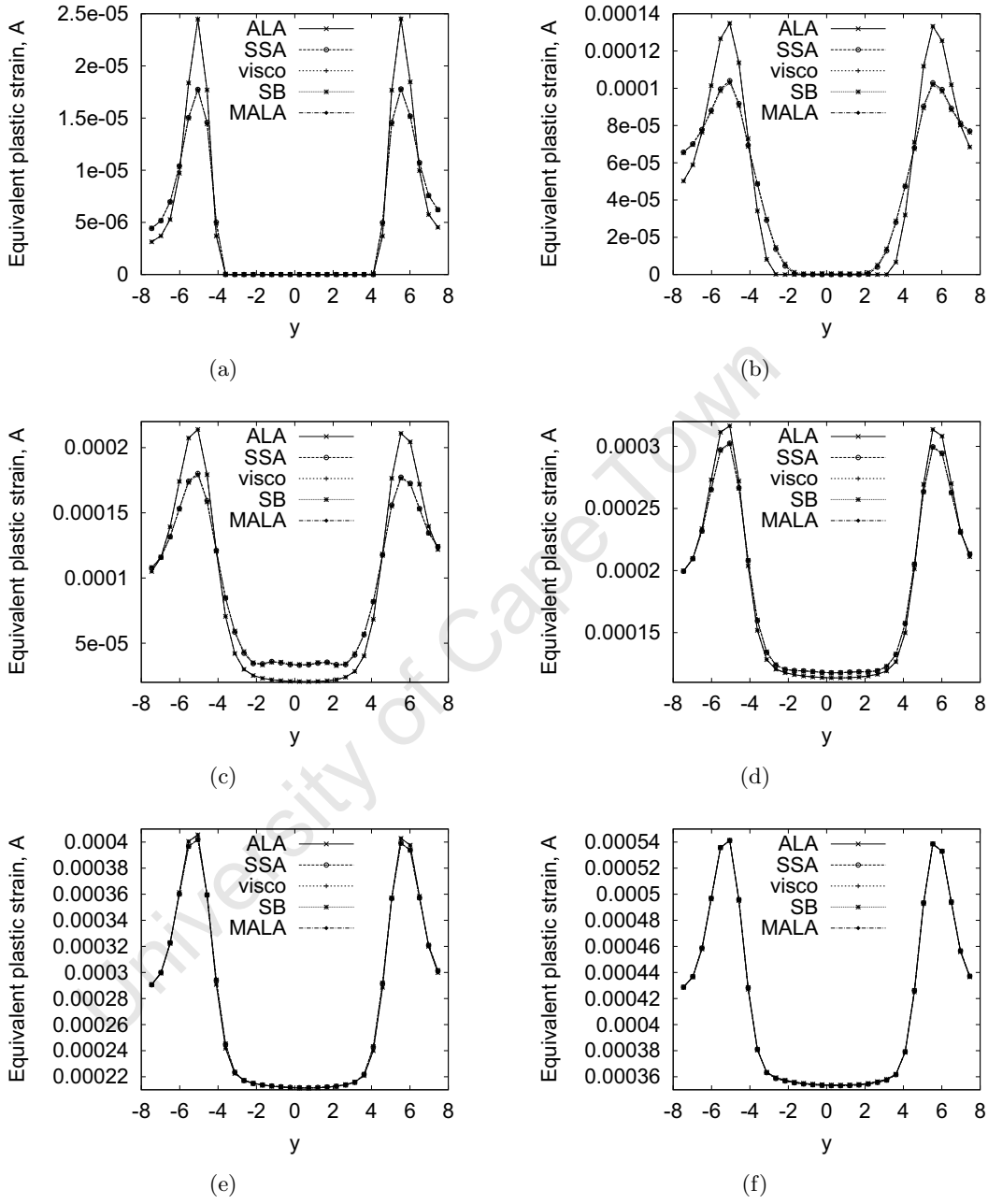


Figure 8.8: The equivalent plastic strains averaged over the elements intersecting the line $(2.7, y, 0)$ at loading parameter (a) $\beta = 101$ (b) $\beta = 102$ (c) $\beta = 103$ (d) $\beta = 105$ (e) $\beta = 107$ (f) $\beta = 110$.

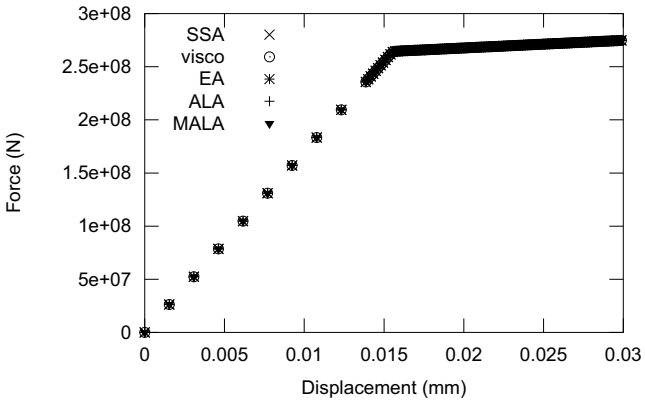


Figure 8.9: The vertical force versus the vertical displacement for the top surface.

	$\theta_1 = 0°$	$\theta_1 = -15°$	$\theta_1 = -30°$
SSA	0.993	0.989	0.990
EA	18.165	18.059	22.263
ALA	7.719	7.700	7.458
MALA	5.960	6.340	6.194

Table 8.3: The total CPU times relative to the VA's total CPU time.

rate-independent algorithm run times by the visocoplastic algorithm time. We use the same setup, boundary conditions and material parameter as described above, except that we set the hardening parameter $q = 1$. This choice of hardening parameter causes an increase in the likelihood of an ill-conditioned system. As in the previous case, the sample is elongated to a load parameter of $\beta = 200$. We choose three different crystal orientations $\{\theta_1, \theta_2, \theta_3\} = \{0°, 0°, 0°\}$, $\{-15°, 0°, 0°\}$, $\{-30°, 0°, 0°\}$. The results are given in Table 8.3. It is clear from this table that the SSA is superior in that it takes much less time than the other rate-independent algorithms, and that it takes slightly less time than the visocoplastic algorithms. We also note that the ALA and MALA performed comparatively better than the EA, but not better than the SSA. The modification to the ALA caused approximately a 17% increase in the efficiency of the algorithm.

For the MALA the success and type of initial guess was recorded, see Table 8.4. Here, $\Delta\lambda^\alpha = \Delta\lambda_n^\alpha, 0$ means that the initial guess $\Delta\lambda^\alpha = \Delta\lambda_n^\alpha$ was used, but later changed to $\Delta\lambda^\alpha = 0$ due to the active set changing. As can be seen the success of the initial guess is prominent, but more importantly scenario where the initial guess is abandoned is low.

Initial guess	$\theta_1 = 0^\circ$	$\theta_1 = -15^\circ$	$\theta_1 = -30^\circ$
$\Delta\lambda^\alpha = \Delta\lambda_n^\alpha$	0.546	0.394	0.452
$\Delta\lambda^\alpha = 0$	0.454	0.606	0.533
$\Delta\lambda^\alpha = \Delta\lambda_n^\alpha, 0$	0	$\ll 0.01$	0.015

Table 8.4: The prevalence of succesful inital guesses for the MALA.

This is a positive result as these scenarios are computationally more expensive than the alternatives.

Various questions are raised by Table 8.3. For instance, what is the cause of the EA time being so much larger? Is it caused by the slow convergence of a few local iterations or in general does the EA take longer to solve each local iteration. In order to get better insight into the reason for the significant difference in the times in Table 8.3 we investigate the time per local iteration. For each quadrature point, we record the time taken for the quadrature point to converge, in other words the time for the local iteration to converge. This is done irrespective of the location of the quadrature point and irrespective of the global Newton–Raphson iteration. In Figure 8.10 we present histograms of the CPU time per local iteration for the various formulations. The histograms in Figures 8.10(a)-8.10(e) clearly indicate that the extended run times are caused by longer local iteration run times. This implies that the EA, ALA, and MALA converge more slowly than the SSA in general. Figures 8.10(c) and 8.10(d) also provided further evidence for the efficiency improvement of the modified ALA (MALA). Figure 8.10(d) illustrates that, as a results of the modification, a large number large number of local iteration run times have become significantly faster.

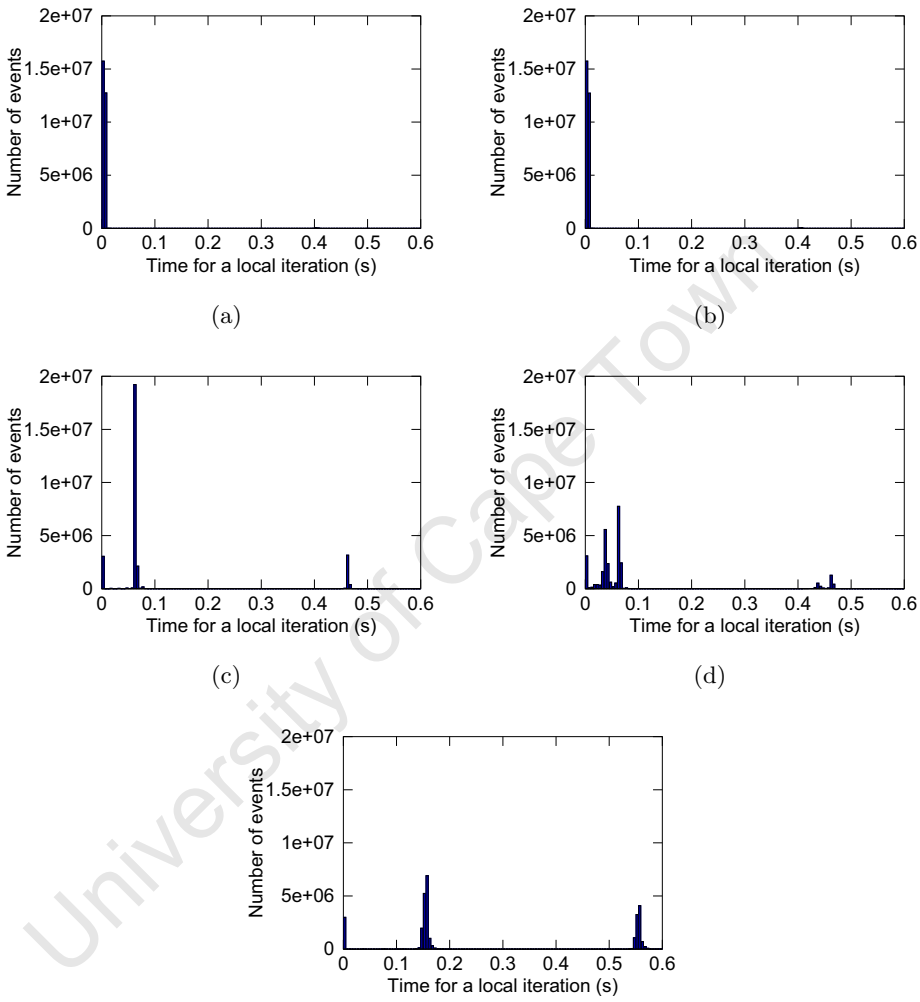


Figure 8.10: Histogram of the CPU time per local iteration for the (a) VA (b) SSA (c) ALA (d) MALA (e) EA.

Chapter 9

Conclusions and recommendations

9.1 Conclusions

The main objective of this thesis was to review and implement various algorithms in rate-independent single-crystal plasticity. The starting point was to justify the mathematical model for single-crystal plasticity. After this was done, various constrained optimisation techniques were applied to the principle of maximum plastic dissipation. Depending on the constrained optimisation technique used, different flow rules were generated. After space and time discretisation, numerical techniques were applied to governing sets of equations. These ultimately resulted in a variety of local algorithms. Finally, these algorithms were applied to two numerical examples to compare agreement and performance.

The first goal was to develop a mathematical model for single crystal plasticity. This involved discussing the relevant theories from continuum mechanics. As a basis for the theory it was relevant to discuss classical plasticity. After this, the response of crystal structures at a microscopic level was discussed to justify further relationships. Adding these relationships to classical plasticity gives rise to single crystal plasticity.

Out of the theory of classical plasticity came the principle of maximum plastic work. This principle can be treated as a constrained maximisation problem. Here it was necessary to give a background of the various mathematical treatments. These were: Lagrange multiplier, penalty, and the augmented Lagrangian formulations. Each application of a formula-

tion to the maximisation problem results to its own set of flow rules.

At this point it was appropriate to discuss the various numerical techniques to be used in the treatment of the problem. This included the numerical linear algebra techniques of LU and singular value decompositions. It is was also relevant to discuss globally convergent Newton–Raphson techniques and the finite element method.

After these numerical methods where introduced, the relevant equations where discretised in space and time. After this, algorithms were developed to solve the various flow rules. In total five algorithms where developed: the SSA, VA, EA, ALA, and MALA. The VA is a rate-dependent algorithm while all the other are rate-independent algorithms. The SSA and the EA were developed from the flow rules derived from the Lagrange multiplier's treatment of the principle of maximum plastic dissipation. The SSA utilises a set search while the EA casts the KKT conditions into a set of equations which is solved for using a Newton–Raphson approach. The ALA and the MALA were developed from the flow rules derived from the augmented Lagrangian treatment of the principle of maximum plastic dissipation. These both apply a fixed point method to the flow rules. The MALA was a modification of the ALA to allow a initial guess based on the previous time iteration.

The implementation of the algorithm in an object orientated frame-work was discussed. Here a sensible class structure was proposed. This class structure serves as a guide and is by no means the only option.

Finally, the algorithms were applied to two problems. The first problem is the shearing of a perfectly plastic single crystal cube. This problem is small, but useful in illustrating the complex behaviour that develops at a quadrature point. Here the slip history results and final shear stress were recorded. The results from the various algorithms compared well with one another and well with results presented in the literature.

The second problem was a tension test. This is a much larger problem, with far more local problems to be solved. The results where compared at macroscopic level, here a good agreement was found for the various algorithms results for the equivalent plastic strain. A more detailed comparison of the equivalent plastic strain was done by plotting the results along the right most boundary of the tension strip. It was found that the results for the algorithms using the augmented Lagrange treatment gave slightly different solutions. The difference in the solution was only prominent in early time steps, and this difference

became small rapidly. It was also noted that at these early time steps the algorithms using an augmented Lagrange treatment had a more localised and pronounced solution.

The tension problem was then used to compare the performance of the various rate-independent algorithms. Here it was found that the SSA greatly out-performed the other algorithms. It was also found that the poorest performing algorithm was the EA, taking approximately twenty times longer than the SSA. The second and third best performing algorithms where the MALA and the ALA. These took approximately 6 and 7.5 times longer than the SSA. The modification to the ALA, found in the MALA, increased the performance by approximately 17%.

Further analysis of the performance results was conducted by monitoring the run times for each quadrature point. This revealed that the increased in performance was related to generally better performance of the local algorithms. In addition, it also shows that poor performance was not caused by an isolated event, but rather by a generally poor performing local algorithm.

Bibliography

[1] deal.ii. http://www.dealii.org/developer/doxygen/deal.II/index.html, October 2010.

[2] iMechanica. http://imechanica.org/node/1394, July 2010.

[3] Paraview. http://www.paraview.org/, October 2010.

[4] Rolls-Royce. http://www.rolls-royce.com/technology_innovation/gas_turbine_tech/turbines.jsp, january 2012.

[5] R. J. Asaro. Micromechanics of crystals and polycrystals. *Advances in applied mechanics*, 23:2–111, 1983.

[6] R. J. Asaro and A. Needleman. Texture development and strain hardening in rate dependent polycrystals. *Acta Metallurgica*, 33(6):923 – 953, 1985.

[7] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II — a general-purpose object-oriented finite element library. *ACM Transactions on Mathematical Software*, 33(4).

[8] C. S. Barrett. *Structure of Metals*. McGraw-Hill, 1943.

[9] M. Becker. *Incompatibility and instability based size effects in crystals and composites at finite elastoplastic strains*. PhD thesis, Stuttgart University, 2006.

[10] B. Moran Belytschko, W. K. Liu. *Nonlinear finite elements for continua and structures*. Wiley, 2000.

[11] M. B. Bettaieb, O. Debordes, A. Dogui, L. Duchêne, and C. Keller. On the numerical integration of rate independent single crystal behavior at large strain. *International Journal of Plasticity*, 2011.

[12] C.C. Celigoj. An augmented lagrange multiplier approach to continuum multislip single crystal thermo-elasto-viscoplasticity. *Communications in Numerical Methods in Engineering*, 21(7):357–376, 2005.

[13] S. Commend and T. Zimmermann. Object-oriented nonlinear finite element programming: a primer. *Advances in Engineering Software*, 32:611–628, 2001.

[14] P. H. Dederichs and G. Leibfried. Elastic Green's function for anisotropic cubic crystals. *Physical Review*, 188(3):1175–1183, Dec 1969.

[15] Y. Dubois-Pelerin and T. Zimmermann. Object-oriented finite element programming: III. An efficient implementation in C++. *Computer Methods in Applied Mechanics and Engineering*, 108:165–183, 1993.

[16] Y. Dubois-Pelerin, T. Zimmermann, and P. Bomme. Object-oriented finite element programming: II. A prototype program in Smalltalk. *Computer Methods in Applied Mechanics and Engineering*, 98:361–397, 1992.

[17] Y. C. Fung and P. Tong. *Classical and computational solid mechanics*. World Scientific, 2001.

[18] W. Han and B. D. Reddy. *Plasticity: mathematical theory and numerical analysis*. Springer, 1999.

[19] W. F. Hosford. *The mechanics of crystals and textured polycrystals*. Oxford Science Publications, 1993.

[20] T. J. R. Hughes. *The finite element method*. Prentice-Hall, 1987.

[21] J. W. Hutchinson. *Proceedings of the Royal Society A*, 319:247, 1970.

[22] W. T. Koiter. Stress-strain relations, uniqueness and variational theorems for elastic-plastic materials with a singular yield surface. *Quarterly of applied mathemeatics*, 11: 350–354, 1953.

[23] E. Kreyszig. *Introductory functional analysis with applications*. Wiley, 1978.

[24] W. M. Lai, D. Rubin, and E. Krempl. *Introduction to continuum mechanics*. Pergamon Press, 1978.

[25] J. Lubliner. On the thermodynamic foundations of non-linear solid mechanics. *International Journal of Non-Linear Mechanics*, 7(3):237 – 254, 1972.

[26] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, 1984.

[27] K. K. Mathur and P. R. Dawson. On modeling the development of crystallographic texture in bulk forming processes. *International Journal of Plasticity*, 5(1):67 – 94, 1989.

[28] C. Miehe. Multisurface thermoplasticity for single crystals at large strains in terms of eulerian vector updates. *International Journal of Solids and Structures*, 33(20-22): 3103 – 3130, 1996.

[29] C. Miehe and J. Schröder. A comparative study of stress update algorithms for rate-independent and rate-dependent crystal plasticity. *International Journal for Numerical Methods in Engineering*, 50:273–298, 2001.

[30] C. Miehe, J.Schröder, and J. Schotte. Computational homogenization analysis in finite plasticity simulation of texture development in polycrystalline materials. *Computer Methods in Applied Mechanics and Engineering*, 171(3-4):387 – 418, 1999.

[31] A. M. Cuiti no and M. Ortiz. Computational modelling of single crystals. *Modeling and Simulation in Material Sciences and Engineering*, 1:225–263, 1992.

[32] J. Pan and J. R. Rice. Rate sensitivity of plastic flow and implications for yield-surface vertices. *International Journal of Solids and Structures*, 19(11):973 – 987, 1983.

[33] D. Peirce, R. J. Asaro, and A. Needleman. Material rate dependence and localized deformation in crystalline solids. *Acta Metallurgica*, 31(12):1951 – 1976, 1983.

[34] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C++: The art of scientific computing*. Cambridge University Press, 2002.

[35] B. D. Reddy. *Introductory functional analysis: with applications to boundary value problems and finite elements*. Springer, 1998.

[36] G. Sarma and T. Zacharia. Integration algorithm for modeling the elasto-viscoplastic response of polycrystalline materials. *Journal of the Mechanics and Physics of Solids*, 47(6):1219 – 1238, 1999.

[37] M. Schmidt-Baldassari. Numerical concepts for rate-independent single crystal plasticity. *Computer Methods in Applied Mechanics and Engineering*, 192(11-12):1261 – 1280, 2003.

[38] P. Seibel. *Coders at work*. Apress, 2009.

[39] J. C. Simo and T. J. R. Hughes. *Computational inelasticity*. Springer, 1994.

[40] P. Steinmann and E. Stein. On the numerical treatment and analysis of finite deformation ductile single crystal plasticity. *Computer Methods in Applied Mechanics and Engineering*, 129(3):235 – 254, 1996.

[41] G. I. Taylor. Plastic strain in metals. *Journal institute of metals*, 62:307, 1938.

[42] K. Terada and I. Watanabe. Computational aspects of tangent moduli tensors in rate-independent crystal elastoplasticity. *Computational Mechanics*, 40(3):497–511, 2007.

[43] L. N. Trefethen and D. Bau III. *Numerical linear algebra*. Siam, 1997.

[44] A. Zamiri, F. Pourboghrat, and F. Barlat. An effective computational algorithm for rate-independent crystal plasticity based on a single crystal yield surface with an application to tube hydroforming. *International Journal of Plasticity*, 23(7):1126 – 1147, 2007.

[45] T. Zimmermann, Y. Dubois-Pelerin, and P. Bomme. Object-oriented finite element programming: I. Governing principles. *Computer Methods in Applied Mechanics and Engineering*, 98:291–303, 1992.