Diploma Thesis

A global local approach for handling local non-linearities

Oikonomakis Lucas

November, 2013

Supervisor: Prof. Manolis Papadrakakis

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF CIVIL ENGINEERING
INSTITUTE OF STRUCTURAL ANALYSIS AND SEISMIC RESEARCH

Zografou Campus
Athens 157 80, Greece

Ευχαριστίες

Η παρούσα διπλωματική εργασία αποτελεί το αποτέλεσμα της πολύμηνης εργασίας μου υπό την επίβλεψη του κ. Μανόλη Παπαδρακάκη, Καθηγητή του Τομέα Δομοστατικής της Σχολής Πολιτικών Μηχανικών ΕΜΠ. Σηματοδοτεί επίσης και το τέλος των προπτυχιακών μου σπουδών στη Σχολή Πολιτικών Μηχανικών του ΕΜΠ, μια σχολή που παρ΄ όλα τα προβλήματα της κατάφερε να προσφέρει υψηλότατο γνωσιακό επίπεδο, αλλά και να σταθεί σαν ένας σημαντικός χώρος κοινωνικοποίησης αυτά τα πέντε χρόνια.

Αρχικά θα ήθελα να ευχαριστήσω τον κ. Παπαδρακάκη. Θέλω να εκφράσω την ευγνωμωσύνη μου απέναντι του, γιατι μου πρόσφερε απλόχερα γνώσεις, αφιέρωσε πολύτιμο χρόνο και με την πείρα του με βοήθησε να ξεπεράσω όποια δυσκολία προέκυπτε. Επίσης, με ενέπνευσε με το ήθος του και τη προσωπικότητα του. Το όραμά του, το πάθος του για τη δουλειά του ήταν αυτά που στάθηκαν σύμμαχοί μου δίνοντας μου όρεξη και διάθεση αυτά τα χρόνια. Έτσι, πιστεύοντας στις ικανότητές μου με καθοδήγησε σε ερευνητικά ζητήματα αιχμής που μου φαίνονταν άπιαστα.

Θα ήθελα ακόμα να ευχαριστήσω τον υποψήφιο Διδάκτορα Αλέξανδρο Καραταράκη που με μύησε στη μαγεία του προγραμματισμού και πάντα κεφάτος με βοήθησε όποτε χρειάστηκε. Ευχαριστώ επίσης και τα υπόλοιπα μέλη της ερευνητικής ομάδας του κ. Παπαδρακάκη που κατα καιρούς συνδράμανε με χρήσιμες συμβουλές και οδηγίες στην εκπόνηση της παρούσας διπλωματικής. Οφείλω επίσης να ευχαριστήσω και τον επιστήθιο φίλο μου Jai Μέξη για τις γλωσσικές του διορθώσεις επί του κειμένου της εργασίας.

Τέλος θα ήθελα να αναφέρω και τα άτομα που βρεθήκανε δίπλα μου όλο αυτον τον καιρό και με την παρουσία τους στην καθημερινότητα μου θεωρώ ότι αποτελούν ένα αναπόσπαστο κομμάτι αυτής της διπλωματικής. Ευχαριστώ το Νίκο για τη συμπόρευσή μας όλο αυτόν το καιρό. Ευχαριστώ επίσης τον Αλέξανδρο, την Τόνια, την Μαριάνθη, τον Κωνσταντίνο για τις όμορφες στιγμες που ζήσαμε. Ολοκληρώνοντας θα ήθελα να ευχαριστήσω την οικογένεια μου που με στηρίζει όλα αυτά τα χρόνια.

Οικονομάκης Λουκάς

Οκτώβρης 2013

Contents

List of Figures

List of Tables

List of Matlab Code

Περίληψη

Η μέθοδος των πεπερασμένων στοιχείων (Finite Element Method) είναι ο βασικός πυλώνας σε θέματα υπολογιστικής μηχανικής τις τελευταίες δεκαετίες. Ωθούμενη απο τις εξελίξεις στην τεχνολογία των υπολογιστών, η μεθοδος των πεπερασμενων στοιχείων αναπτύσεται με μεγάλη ταχύτητα τόσο σε ακαδημαϊκό όσο και σε εργασιακό επίπεδο. Από την πληθώρα τεχνολογιών, σχετικών με τα πεπερασμενα στοιχεία, που έχουν αναπτυχθεί, η παρούσα διπλωματική εργασία επαφίεται σε θέματα υποφορέων και πολυεπίπεδης ανάλυσης.

Οι διάφορες τεχνικές υποφορέων είναι ισχυρότατα εργαλεία σε θέματα υπολογιστικής μηχανικής. Χωρίζοντας ενα μεγάλο πρόβλημα σε μικρότερα, οι τεχνικές αυτές καταφέρνουν να κρατούν το υπολογιστικό κόστος σε διαχειρίσιμα επίπεδα. Παρόλα αυτα η επιτυχία τους εξαρτάται απο τον τρόπο σύνδεσης αυτών των υποπροβλημάτων. Απο την άλλη, σε περιπτώσεις όπου συναντάμε χωρικές και χρονικές ανομοιότητες χρησιμοποιούνται τεχνικές πολυεπίπεδης ανάλυσης.

Πολύ συχνά κατά την εφαρμογή της μεθόδου ερχόμαστε αντιμέτωποι με προβλήματα στα οποία ολόκληρος ο φορέας συμπεριφέρεται γραμμικά-ελαστικά με εξαίρεση μια πολύ μικρή περιοχή όπου η συμπεριφορά του είναι μη γραμμική. Συνήθως, σε ένα τέτοιο προβλημα χωρίζουμε τη χρονοιστορία φόρτισης σε βηματικά επιβαλλόμενα φορτία και λύνουμε μη γραμμικά για το κάθε φορτίο. Αν όμως το πρόβλημα είναι πολύ μεγάλο, αυτή η προσέγγιση είναι ασύμφορη απο άποψη υπολογιστικού κόστους.

Στην παρούσα διπλωματική εργασία προτείνεται μια μέθοδος για την επίλυση τέτοιων φαινομένων, όπου μη-γραμμικά φαινόμενα εμφανίζονται σε μια μικρή περιοχή του φορέα ενώ ο υπόλοιπος παραμένει γραμμικός και ελαστικός. Για το σκοπό αυτό δύο μοντέλα πεπερασμένων στοιχείων δημιουργούνται. Ένα καθολικό που προσομοιάζει ολόκληρο το φορέα με γραμμικές ελαστικές ιδιότητες, και ένα τοπικό μη-γραμμικό που αντικαθιστά τον καθολικό στη μη-γραμμική περιοχή. Με μια επαναληπτική διαδικασία επιτυγχάνεται η ακριβής επίλυση του προβλήματος.

Σε αυτήν τη μεθοδολογία καλούμαστε να υπολογίσουμε την μηχανική στιβαρότητα μιας περιοχής, η οποία περιγράφεται από το συμπλήρωμα του Schur του μητρώου δυσκαμψίας της περιοχής πάνω στο σύνορο. Αυτή η στιβαρότητα μπορεί να υπολογιστεί απευθείας με μια στατική συμπύκνωση ή εναλλακτικά να προσεγγιστεί με κάποιον πιο εύκολο τρόπο, καθώς ο απευθείας υπολογισμός έχει μεγάλο κοστος για μεγαλα προβλήματα. Αυτή η προσέγγιση επιτυγχάνεται συνδιάζοντας τεχνικές μικρο-κλίμακας (λωρίδες στοιχείων) και μεγα-κλίμακας (τεχνικές ομογενοποίησης). Επίσης, αυτή η εναλλακτική είναι μη παρεμβατική ως προς το πρόγραμμα πεπερασμένων που έχει τα δύο μοντέλα, που σημαίνει ότι τα μοντέλα δεν αλλάζουν και υπολογισμοί μπορούν να γίνουν απο έτοιμα προγραμματα πεπερασμένων στοιχείων.

Διάφοροι τρόποι ανταλλαγής δεδομένων μεταξύ των μοντέλων παρουσιάζονται καθώς και μια μέθοδος για την επίλυση περιπτώσεων όπου τα πλέγματα των δύο μοντέλων είναι διαφορετικά ακόμα και στο σύνορο. Τέλος, οι ιδιότητες των μεθόδων μελετώνται με δύο παραδείγματα.

Η σημασία της μεθόδου είναι ότι, εκτός από τοπικές μη γραμμικότητες μπορεί να χρησιμοποιηθεί και για την εισαγωγή γεωμετρικών ατελειών, διαφορετικών πλεγμάτων ή καταστατικών νόμων που λείπουν από το καθολικό μοντέλο. Το τοπικό μοντέλο μπορεί επίσης να αναλυθεί σε τελείως διαφορετικό λογισμικό, το οποίο μπορεί να περιέχει δυνατότητες που δεν περιέχει το λογισμικό του καθολικού μοντέλου. Έτσι, η μέθοδος αυτή μπορεί να αποτελέσει ένα ισχυρό εργαλείο για την επικοινωνία διφορετικών τύπων πεπερασμένων στοιχείων και για την ακριβέστερη ανάλυση παλιότερων μοντέλων ή φορέων που έχουν υποστεί τροποποιήσεις.

Abstract

The Finite Element Method (FEM) has been the dominant technique in computational mechanics in the past decades. Assisted by the advances in computers, FEM is developing with great speed in both academia and industry. Among the plethora of the methods that have been developed, the current thesis puts it's focus on a subject that is related to substructuring and multiscale analysis.

Substructuring and domain decomposition methods are very powerful analysis techniques in the field of structural mechanics. By splitting a large problem into several smaller subproblems, these techniques can help keeping computational costs at reasonable levels. However, their efficiency depends entirely on how well the subproblems are bridged together. On the other hand, wherever large disparities in spatial and temporal scales are encountered the simulation efforts are dominated by multiple scales. This is where multiscale analysis is used.

In the aircraft industry, it is a common task to perform a finite element analysis on a complex structure that mostly evolves in a linear elastic way, but exhibits confined plasticity (or other nonlinear phenomena) in a small critical region. In most FE software, such an analysis is usually carried out by dividing the loading history into several load increments, and by solving nonlinear equilibrium equations at each increment, using Newton's method or one of its variants. When the problem size is too large or the loading history is too complex, this approach can lead to unaffordable computational costs.

This thesis proposes a computational strategy to solve such structural problems, where non-linear phenomena occur within a small area, while the rest of the structure retains a linear elastic behaviour. Two finite element models are defined: a global linear model of the whole structure, and a local non-linear 'sub-model' meant to replace the global model in the non-linear area. An iterative coupling technique is then used to perform this replacement in an exact way.

In this technique we are called to compute the 'mechanical impedance' of a region that can be described by the Schur complement of its stiffness matrix on its boundary. This quantity can be computed from a static condensation (which basically consists in computing it straightforwardly) or approximated in a second non-intrusive way, as the first way is usually very expensive on large problems. This approximation of the Schur complement of a subdomain's stiffness matrix is obtained by combining local (i.e. element strips) and global (i.e. homogenized) contributions. Furthermore, this variation is non-intrusive, which means the model data sets are never modified and the computations can be carried out with standard finite element software.

Several ways of exchanging data between the models are discussed and a simple solution is introduced for the handling of non-conforming meshes. Finally, the properties of the methods are investigated on two examples.

The significance of this method is that except from local non-linearities it can be used to introduce geometric details, mesh refinements or specific constitutive laws that are absent from the global model. The local model can also be analysed using a separate piece of code, which may contain features that are not implemented in the global Finite Element solver. In that way, it can be a powerful tool for the connection of different types of elements, for more detailed analyses of existing models and for reanalysis after model modifications.

Συνοπτική περιγραφή της διπλωματικής στα ελληνικά

Περιγραφή του προβλήματος

Στις μέρες μας υπάρχει μια ραγδαία ανάπτυξη της μεθόδου των πεπερασμένων στοιχείων. Συνεχώς τόσο σε ακαδημαϊκό όσο και εργασιακό επίπεδο, η μέθοδος των πεπερασμένων στοιχείων διδάσκεται, χρησιμοποιείται και αναπτύσεται. Νεα δεδομένα, νεες τεχνολογίες εμφανίζονται συνεχώς δείχνοντας μας ότι πρόκειται για μια επιστήμη με λαμπρό μέλλον.

Πολύ συχνά κατά την εφαρμογή της μεθόδου ερχόμαστε αντιμέτωποι με προβλήματα στα οποία ολόκληρος ο φορέας συμπεριφέρεται γραμμικά-ελαστικά με εξαίρεση μια πολύ μικρή περιοχή όπου η συμπεριφορά του είναι μη γραμμική. Είναι πολύ πιθανόν η συμπεριφορά αυτής της μη γραμμικής περιοχής να είναι και κρίσιμη για την συμπεριφορά ολόκληρου του φορέα (μετατοπίσεις, δυνάμεις). Η μη γραμμική ανάλυση ολόκληρης της κατασκευής αν και αποτελεί λύση είναι πολλές φορές ασύμφορη από αποψη χρόνου (σε περιπτώσεις που ο φορέας είναι μεγάλος).

Στην παρούσα διπλωματική εργασία παρουσιάζεται μια μέθοδος επίλυσης του παραπάνω προβλήματος. Κρατάμε το μοντέλο του φορέα με γραμμικές-ελαστικές ιδιότητες (όπως ήταν δηλαδή) και δημιουργούμε ένα δεύτερο μοντέλο μόνο για την περιοχή που μας ενδιαφέρει με τις μη γραμμικές ιδιότητες που την χαρακτηρίζουν. Η μέθοδος που χρησιμοποιήθηκε είναι μια επαναληπτική μέθοδος που στηρίζεται στη μεταφορά δεδομένων μεταξύ των δύο μοντέλων (δυνάμεις, μετατοπίσεις) και με αλλεπάλληλες επιλύσεις του καθενός πετυχαίνει την τελική σύγκληση και επίτευξη λύσης.

Καθώς υπάρχει μεταφορά πληροφοριών μεταξύ των δύο μοντέλων η τελική λύση πρόκειται για ακριβή λύση στο πρόβλημα. Δηλαδή η τελική λύση λαμβάνει υπόψιν της όλες τις ανακατανομές των τάσεων που συμβαίνουν λόγω της μη γραμμικότητας και η λύση που λαμβάνουμε θα ήταν η ίδια με το να αντικαταστούσαμε την συγκεκριμένη περιοχή με μη γραμμικά πεπερασμένα.

Στην παρούσα διπλωματική παρουσιάζονται δύο παραλλαγές της μεθόδου.

Μία από αυτές έχει και την ιδιαιτερότητα ότι είναι μη παρεμβατική ως προς το πρόγραμμα πεπερασμένων που έχει τα δύο μοντέλα. Αυτό σημαίνει ότι τα δύο μοντέλα πεπερασμενων στοιχείων μπορούν να βρίσκονται σε οποιοδήποτε πρόγραμμα πεπερασμένων (ακόμα και κλειστό πρόγραμμα π.χ. Abacus). Πρακτικά, οι διαδικασίες της μεθόδου που προτείνεται δεν απαιτούν από το πρόγραμμα στο οποίο βρίσκονται τα μοντέλα πληροφορίες που δεν θα μπορούσε να επιστρέψει (π.χ. μητρώα δυσκαμψίας) ούτε επεμβαίνουν με τροποποίηση των μοντέλων ή των μεθόδων επίλυσης που διαθέτει.

Φυσικά η μέθοδος επίλυσης του παραπάνω προβλήματος μπορεί να χρησιμοποιηθεί για την επίλυση πολλών παρόμοιων προβλημάτων. Δηλαδή το δεύτερο μοντέλο που κατασκευάζεται για να προσωμιάσει μια περιοχή του φορέα μπορεί να έχει οποιαδήποτε ιδιαιτερότητα. Έτσι στο δεύτερο μοντέλο μπορούμε να έχουμε διαφορετική γεωμετρία, οπές ή ρωγμές, διαφορετικούς καταστατικούς νόμους για το υλικό, μη γραμμικότητα γεωμετρίας ακόμα και διαφορετικά πεπερασμένα στοιχεία με τελέιως διαφορετικό δίκτυο.

Περιγραφή της γενικής μεθοδολογίας

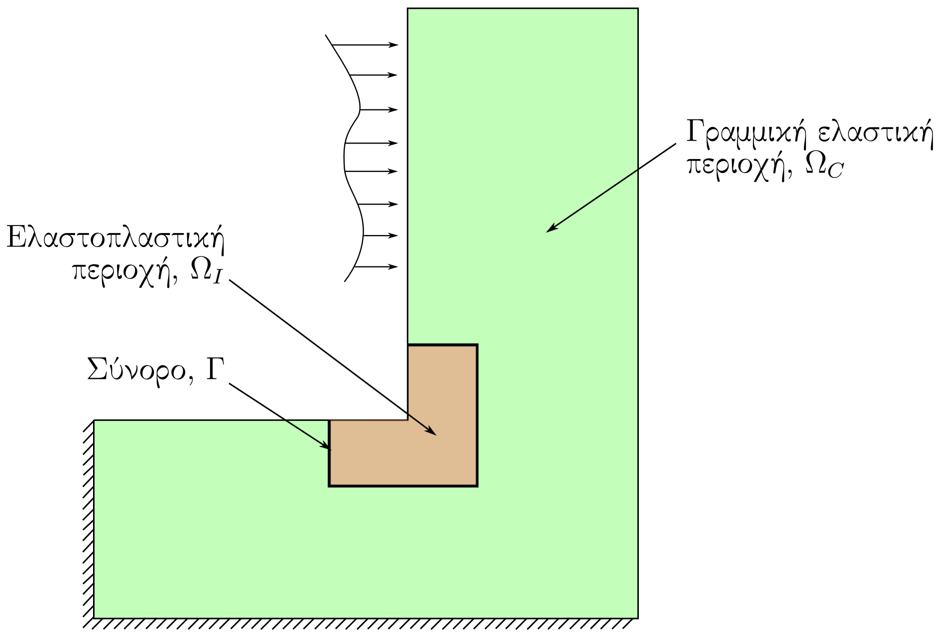
Θεωρούμε ενα μηχανικό στατικό πρόβλημα στο πεδίο Ω. Επίσης θεωρούμε οτι η συμπεριφορά ολόκληρου του πεδίου είναι γραμμική και ελαστική εκτός από μια μικρή περιοχή, που συμβολίζουμε Ω_I. Σε αυτή την περιοχή θεωρούμε ότι ο καταστατικός νόμος του υλικού είναι ελαστοπλαστικός. Επιπλέον, θεωρούμε ότι η περιοχή Ω_I, όπου η ελαστοπλαστική συμπεριφορά εμφανίζεται είναι γνωστή από πριν, και ότι η υπόλοιπη περιοχή, που θα αναφέρεται ώς συμπληρωματική περιοχή, $\Omega_C = \Omega \backslash \Omega_I$, θα παραμείνει ελαστική και γραμμική. Το παραπάνω πρόβλημα φαίνεται στο Σχήμα 1.

Το πρόβλημα μπορεί να επαναδιατυπωθεί χρησιμοποιώντας δύο μοντέλα, το καθολικό μοντέλο και το τοπικό μοντέλο. Το καθολικό μοντέλο είναι ένα γραμμικό ελαστικό μοντέλο (Σχήμα 2α΄). Αντιπροσωπεύει ολόκληρο το φορέα και τη μη γραμμική περιοχή αλλά με γραμμικές ελαστικές ιδιότητες. Το τοπικό μοντέλο περιγράφει μόνο την μη γραμμική περιοχή (Σχήμα 2β΄). Σε αυτό το σημείο πρέπει να διευκρινιστεί ότι:

1. το Ω_C είναι ένα σύνολο πεπερασμένων στοιχείων που δεν χωρίζεται από το καθολικό πλέγμα (το σύνορο Γ δεν κόβει κανένα στοιχείο του καθολικού μοντέλου)

2. η Ω_I είναι η ακριβής περιοχή που καλύπτεται από το τοπικό πλέγμα, και το Γ είναι μέρος του συνόρου του

3. και τα δύο πλέγματα ταυτίζονται στο Γ. Δηλαδή έχουν ίδιους κόμβους,


Γραμμική ελαστική περιοχή, Ω_C
Ελαστοπλαστική περιοχή, Ω_I
Σύνορο, Γ

Σχήμα 1: Το πρόβλημα αναφοράς

βαθμούς ελευθερίας, συναρτήσεις σχηματος κλπ. Οπότε δεν χρειάζεται διαχωρισμός του Γ.

4. τα δύο μοντέλα επηρεάζουν το ένα το άλλο μόνο στο σύνορο Γ. Οπότε δεν χρειάζεται να υπάρχει συμβατότητα στα πλέγματα των δύο μοντέλων στο Ω_I.

Η λογική της μεθοδολογίας είναι ότι η τελική λύση λαμβάνει υπόψιν της το τοπικό μοντέλο για τη μη γραμμική περιοχή και το καθολικό μοντέλο για την υπόλοιπη περιοχή (Σχήμα 3). Οπότε το πεδίο τω μετατοπίσεων είναι

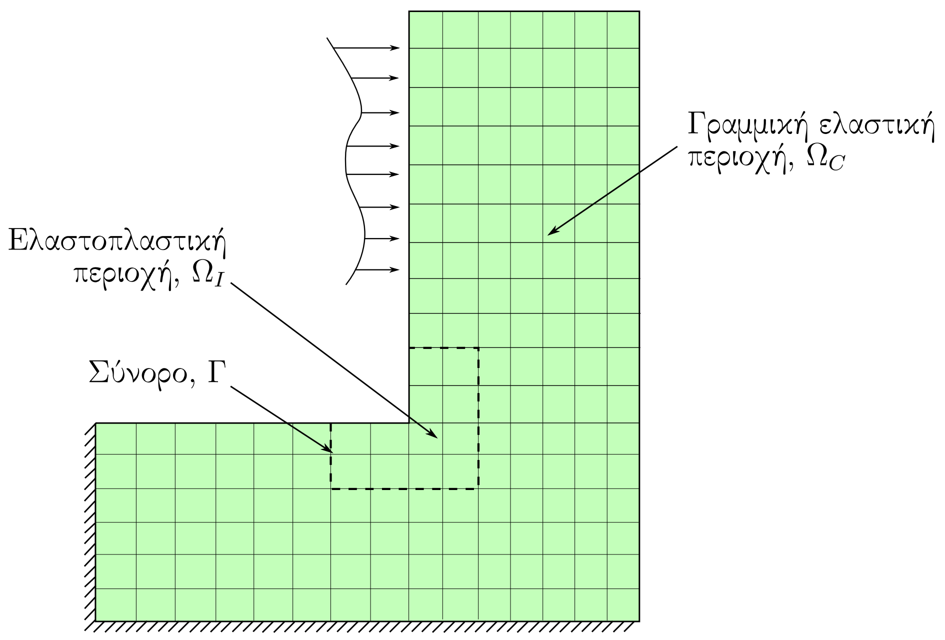
$$\mathbf{u} = \begin{cases} \mathbf{u}^L(\mathbf{x}) & \text{αν } \mathbf{x} \in \Omega_I \\ \mathbf{u}^G(\mathbf{x}) & \text{αν } \mathbf{x} \in \Omega_C \end{cases}$$

ενώ το πεδίο των τάσεων, $\boldsymbol{\sigma}$, είναι

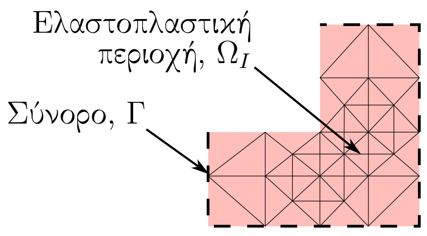
$$\boldsymbol{\sigma} = \begin{cases} \boldsymbol{\sigma}^L(\mathbf{x}) & \text{αν } \mathbf{x} \in \Omega_I \\ \boldsymbol{\sigma}^G(\mathbf{x}) & \text{αν } \mathbf{x} \in \Omega_C \end{cases}$$

Η τελική λύση λοιπόν θα πρέπει να ικανοποιεί τις παρακάτω συνθήκες:

1. η τοπική λύση, $\mathbf{u}^L$ και $\boldsymbol{\sigma}^L$, πρέπει να ικανοποιεί κάθε εξίσωση του Ω_I και του $\partial\Omega_I \backslash \Gamma$ (εξισώσεις ισορροπίας, συνοριακές συνθήκες και ελαστοπλαστικές καταστατικές εξισώσεις του υλικού)


Γραμμική ελαστική περιοχή, Ω_C
Ελαστοπλαστική περιοχή, Ω_I
Σύνορο, Γ

(α΄) Καθολικό μοντέλο

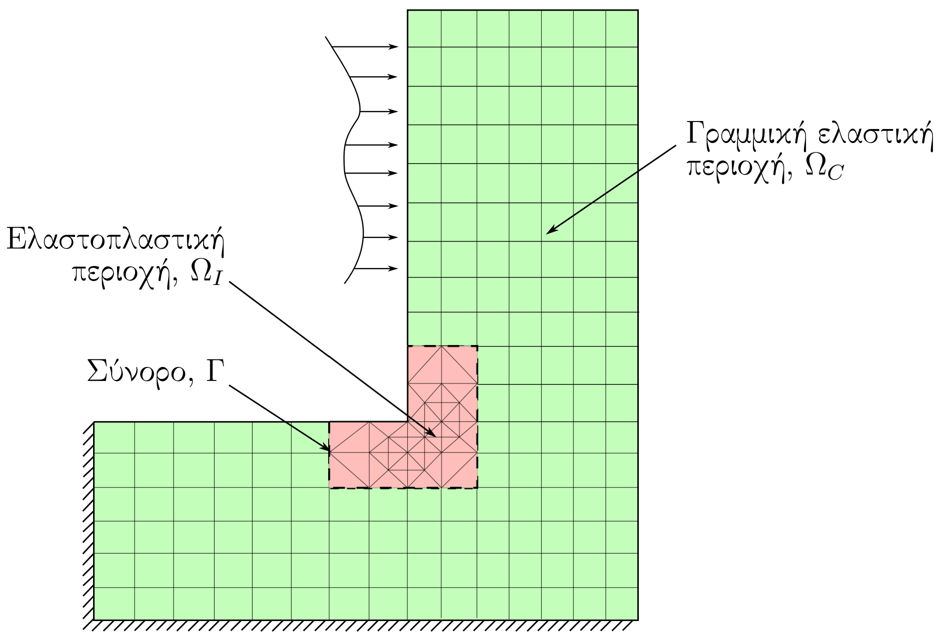


(β΄) Τοπικό μοντέλο

Σχήμα 2: Τα δύο μοντέλα της μεθοδολογίας

2. η καθολική λύση περιορισμένη στο Ω_C, $\mathbf{u}_C^G$ και $\boldsymbol{\sigma}_C^G$, πρέπει να ικανοποιεί κάθε εξίσωση του Ω_C και του $\partial\Omega_C \backslash \Gamma$ (εξισώσεις ισορροποίας, συνοριακές συνθήκες και γραμμικές ελαστικές καταστατικές εξισώσεις του υλικού)

3. και οι δύο λύσεις πρέπει να ταυτίζονται στο Γ (κοινές μετατοπισεις και ισορροπία εσωτερικών δυνάμεων)


Γραμμική ελαστική περιοχή, Ω_C
Ελαστοπλαστική περιοχή, Ω_I
Σύνορο, Γ

Σχήμα 3: Το μοντέλο αναφοράς

Μεθοδολογία επίλυσης - Χρήση μετατοπίσεων

Η επίλυση ξεκινάει με μια αρχική γραμμική ανάλυση του καθολικού φορέα. Η μεθοδολογία συνεχίζει ως εξής:

1. **Τοπική ανάλυση.** Οι μετατοπίσεις του συνόρου Γ, που έχουν υπολογιστεί από την καθολική ανάλυση, μεταφέρονται στο σύνορο του τοπικού μοντέλου και εκτελείται μια ανάλυση ελέγχου μετακινήσεων (Displacement control analysis).
$$\mathbf{u}_\Gamma^L = \mathbf{u}_\Gamma^G$$

2. **Υπολογισμός υπολοιπόμενων δυνάμεων.** Καθώς με την προηγούμενη ανάλυση η συνέχεια των μετακινήσεων έχει επιτευχθεί, οι υπολοιπόμενες δυνάμεις υπολογίζονται λαμβάνοντας υπόψιν την έλλειψη ισορροπίας στις εσωτερικές δυνάμεις για το σύνορο Γ. Οπότε υπολογίζονται ως εξης
$$\mathbf{r}_\Gamma = -(\boldsymbol{\lambda}_\Gamma^L + \boldsymbol{\lambda}_{\Gamma,C}^G)$$
όπου $\boldsymbol{\lambda}_\Gamma^L$ είναι οι εσωτερικές δυνάμεις του τοπικού μοντελου για τους κόμβους του συνόρου Γ και $\boldsymbol{\lambda}_{\Gamma,C}^G$ αντίστοιχα οι εσωτερικές δυνάμεις του καθολικού μοντέλου.

3. **Διόρθωση της καθολικής λύσης.** Το καθολικό μοντέλο φορτίζεται με τις υπολοιπόμενες δυνάμεις και μόνο. Από την γραμμική ανάλυση του καθολικού μοντέλου προκύπτει ο διορθωτικός όρος $\Delta\mathbf{u}^G$ ο οποίος αντιπροσωπεύει την συμβολή της μη γραμμικής περιοχής στις μετατοπίσεις του καθολικού μοντέλου.

$$\mathbf{K}^G\Delta\mathbf{u}^G = \mathbf{r}^G = \begin{bmatrix} 0 \\ \mathbf{r}_\Gamma \\ 0 \end{bmatrix}$$

   και η καθολική λύση διορθώνεται ως εξής

$$\mathbf{u}^G \leftarrow \mathbf{u}^G + \Delta\mathbf{u}^G$$

Μεθοδολογία επίλυσης - Χρήση μικτών συνοριακών συνθηκών

Αναφερόμενοι σε επίλυση με τη χρήση μικτών συνοριακών συνθηκών εννοούμε ότι για την επίλυση του τοπικού μοντέλου δεν μεταφέρουμε τις μετατοπίσεις του καθολικού (όπως στη προηγούμενη μεθοδολογία) ούτε χρησιμοποιούμε την ισορροπία των εσωτερικών δυνάμεων (που θα ήταν μια άλλη παραλλαγή) αλλά ένα γραμμικό συνδιασμό αυτών των δύο. Οπότε

$$(\boldsymbol{\lambda}_\Gamma^L + \boldsymbol{\lambda}_{\Gamma,C}^G) + \mathbf{A}(\mathbf{u}_\Gamma^L - \mathbf{u}_\Gamma^G) = 0$$

όπου $\mathbf{A}$ είναι μια παράμετρος της μεθόδου, αλλά επίσης ένας τετράγωνος πίνακας που αντιπροσωπεύει την στιβαρότητα του συνόρου. Αντίστοιχα οι υπολοιπόμενες δυνάμεις υπολογίζονται ως εξής

$$(\boldsymbol{\lambda}_\Gamma^L + \boldsymbol{\lambda}_{\Gamma,C}^G) - \mathbf{B}(\mathbf{u}_\Gamma^L - \mathbf{u}_\Gamma^G) = 0$$

όπου $\mathbf{B}$ είναι επίσης μια παράμετρος της μεθόδου. Η επιλογή του $\mathbf{A}$ και του $\mathbf{B}$ θα αναφερθεί αργότερα. Οπότε η μεθοδολογία σε αυτή την παραλλαγή είναι η εξής:

1. **Τοπική ανάλυση.** Χρησιμοποιώντας τις μικτές συνοριακές συνθήκες η εξίσωση της ανάλυσης του τοπικού μοντέλου παίρνει την μορφή

$$\begin{bmatrix} \mathbf{g}_\Gamma^L(\mathbf{u}_\Gamma^L, \mathbf{u}_I^L) \\ \mathbf{g}_I^L(\mathbf{u}_\Gamma^L, \mathbf{u}_I^L) \end{bmatrix} + \begin{bmatrix} \mathbf{A} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_\Gamma^L \\ \mathbf{u}_I^L \end{bmatrix} = \begin{bmatrix} \mathbf{f}_\Gamma^L \\ \mathbf{f}_I^L \end{bmatrix} + \begin{bmatrix} -\boldsymbol{\lambda}_{\Gamma,C}^G + \mathbf{A}\mathbf{u}_\Gamma^G \\ 0 \end{bmatrix}$$

   Είναι φανερό ότι ο πίνακας $\mathbf{A}$ είναι μια προσθήκη στιβαρότητας στο σύνορο ενώ η ποσότητα $-\boldsymbol{\lambda}_{\Gamma,C}^G+\mathbf{A}\mathbf{u}_\Gamma^G$ είναι το επιπρόσθετο φορτίο στο σύνορο.

Είναι εύκολο εδώ να παρατηρηθεί ότι **ο πίνακας A αντιπροσωπεύει την στιβαρότητα του υπόλοιπου φορέα** οπότε και κάνει την χρήση των μικτών συνοριακών συνθηκών μια πιο ρεαλιστική προσέγγιση από τη χρήση των μετατοπίσεων.

2. **Υπολογισμός υπολοιπόμενων δυνάμεων.** Οι υπολοιπόμενες δυνάμεις υπολογίζονται ώς εξής

$$\mathbf{r}_\Gamma = -(\boldsymbol{\lambda}_\Gamma^L + \boldsymbol{\lambda}_{\Gamma,C}^G) + \mathbf{B}(\mathbf{u}_\Gamma^L - \mathbf{u}_\Gamma^G)$$

   η χρησιμοποιώντας την ισορροπία που περιέχει τον πίνακα $\mathbf{A}$

$$\mathbf{r}_\Gamma = [\mathbf{A} + \mathbf{B}](\mathbf{u}_\Gamma^L - \mathbf{u}_\Gamma^G)$$

3. **Διόρθωση της καθολικής λύσης.** Όπως και πριν το καθολικό μοντέλο φορτίζεται με τις υπολοιπόμενες δυνάμεις και μόνο. Πραγματοποιείται γραμμική ανάλυση στο καθολικό μοντέλο και οι μετατοπίσεις διορθώνονται με τον ίδιο τρόπο.

Υπολογισμός του συμπληρώματος του Schur

Στη προηγούμενη ενότητα είδαμε ότι για τις μικτές συνοριακές συνθήκες ισχύει

$$(\boldsymbol{\lambda}_\Gamma^L + \boldsymbol{\lambda}_{\Gamma,C}^G) + \mathbf{A}(\mathbf{u}_\Gamma^L - \mathbf{u}_\Gamma^G) = 0$$

Επίσης ξέρουμε ότι για το τοπικό μοντέλο ισχύει η εξίσωση

$$\mathbf{h}_\Gamma^L(\mathbf{u}_\Gamma^L) = \boldsymbol{\lambda}_\Gamma^L$$

όπου $\mathbf{h}_\Gamma^L = \mathbf{h}_\Gamma^L(\mathbf{u}_\Gamma^L)$ είναι η μη γραμμική σχέση που συνδέει τις μετατοπίσεις με τις δυνάμεις για το τοπικό πρόβλημα. Για το συμπληρωματικό φορέα ισχύει

$$\mathbf{S}_{\boldsymbol{\Gamma},\mathbf{C}}^{\mathbf{G}}\mathbf{u}_\Gamma^G = \mathbf{b}_{\Gamma,C}^G + \boldsymbol{\lambda}_{\Gamma,C}^G$$

όπου $\mathbf{S}_{\boldsymbol{\Gamma},\mathbf{C}}^{\mathbf{G}}$ είναι το συμπλήρωμα του Schur του συμπληρωματικού φορέα πάνω στο σύνορο. Το συμπλήρωμα του Schur βρίσκεται από στατική συμπύκνωση όλων των υπολοίπων βαθμών ελευθερίας ώστε να μείνουν μόνο αυτοί του συνόρου. Τέλος το σύνορο θα πρέπει να ικανοποιεί τις εξής εξισώσεις

$$\begin{aligned} \mathbf{u}_\Gamma^G &= \mathbf{u}_\Gamma^L \\ \boldsymbol{\lambda}_{\Gamma,C}^G + \boldsymbol{\lambda}_\Gamma^L &= 0 \end{aligned}$$

Συνδιάζοντας τα παραπάνω η λύση του προβλήματος αναφοράς παίρνει την μορφή

$$\mathbf{h}_\Gamma^L(\mathbf{u}_\Gamma^R) + \mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}\mathbf{u}_\Gamma^R = \mathbf{b}_{\Gamma,C}^G$$

ενώ χρησιμοποιώντας τις δύο πρώτες εξισώσεις αυτού του κεφαλαίου παίρνουμε

$$\mathbf{h}_\Gamma^L(\mathbf{u}_\Gamma^L) + \mathbf{A}\mathbf{u}^L = \mathbf{A}\mathbf{u}^G - \boldsymbol{\lambda}_C^G$$

Με άλλα λόγια η δυσκαμψία του συνόρου $\mathbf{A}$ μεταφέρεται στο τοπικό πρόβλημα, το οποίο φορτίζεται και με το μικτό φορτίο $\mathbf{A}\mathbf{u}^G - \boldsymbol{\lambda}_C^G$. Από τις δύο τελευταίες εξισώσεις φαίνεται ότι αν το μητρώο $\mathbf{A}$ είναι ίσος με το συμπλήρωμα του Schur για τον συμπληρωματικό φορέα, $\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}$, τότε το τοπικό προβλημα ταυτίζεται με τη λύση του προβλήματος αναφοράς. Οπότε αν υπολογίσουμε το $\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}$ ακριβώς, η μεθοδολογία μας θα είναι:

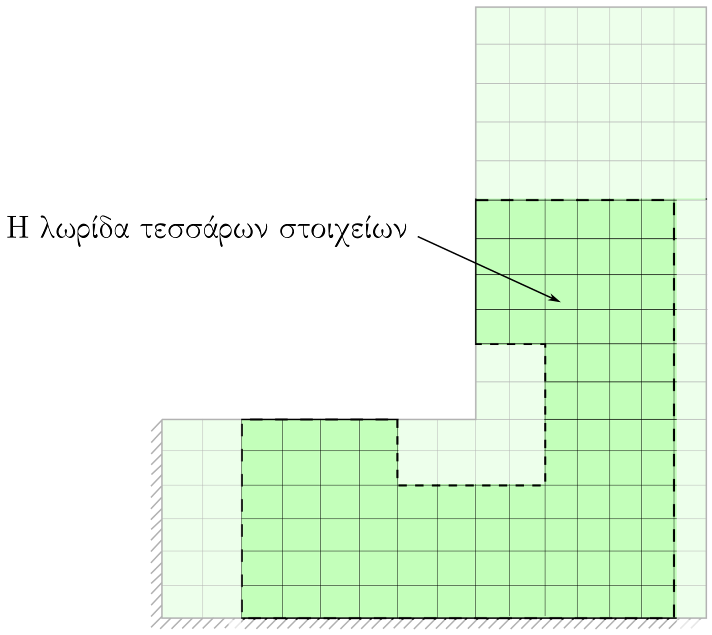
1. ακριβής, δηλαδή η τοπική λύση θα ταυτίζεται με τη λύση αναφοράς
2. άμεση, δηλαδή με την πρώτη ανάλυση του τοπικού μοντέλου θα φτάναμε και στην λύση του προβλήματος χωρίς να χρειάζονται άλλες επαναλήψεις
3. αδιάφορη ως προς τη θέση του συνόρου

Η καλύτερη λοιπόν επιλογή για τον πίνακα $\mathbf{A}$ είναι ο πίνακας $\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}$. Αυτός ο πίνακας θα προκύψει μετά από στατική συμπύκνωση όλων των βαθμών ελευθερίας πλην των βαθμών ελευθερίας του συνόρου για τον συμπληρωματικό φορέα. Η λύση αυτή αν και εφικτή είναι γνωστό ότι είναι ιδιαίτερα προβληματική για μεγάλους φορείς λόγω του υπολογιστικού της κόστους. Επίσης χρειάζεται και την δημιουργία του συμπληρωματικού φορέα που είναι μια επεμβατική διαδικασία στο καθολικό μοντέλο, καθώς πρέπει να διαγραφούν τα στοιχεία που βρίσκονται στην μη γραμμική περιοχή.

Προτείνεται λοιπόν και μια δεύτερη εναλλακτική για την λύση του παράνω προβλήματος. Αντί να υπολογιστεί το $\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}$ απευθείας να υπολογιστεί μια προσέγγιση του. Η προσέγγιση αυτή θα πρέπει να είναι κοντά στην ακριβή τιμη αλλά και με μικρότερο υπολογιστικό κόστος. Διάφορες προσεγγίσεις έχουν προταθεί κατα καιρούς. Κάποιες βασίζονται στα στοιχεία κοντά στο σύνορο οπότε θα τις αποκαλούμε προσεγγίσεις μικρο-κλίμακας ενώ οι υπόλοιπες εμπνευσμένες από τεχνικές ομογενοποίησης θα τις αποκαλέσουμε προσεγγίσεις μεγα-κλίμακας. Στη παρούσα διπλωματική εφαρμόστηκε ένας συνδιασμός των παραπάνω προσεγγίσεων που προτάθηκε από τους Genre et. al [7].

Προσέγγιση μικρο-κλίμακας

Σε αυτή την προσέγγιση αντί να γίνει στατική συμπύκνωση ολόκληρου του φορέα, γίνεται στατική συμπύκνωση σε μια λωρίδα στοιχείων γύρω από το σύνορο (Σχήμα 4). Η διαδικασία είναι



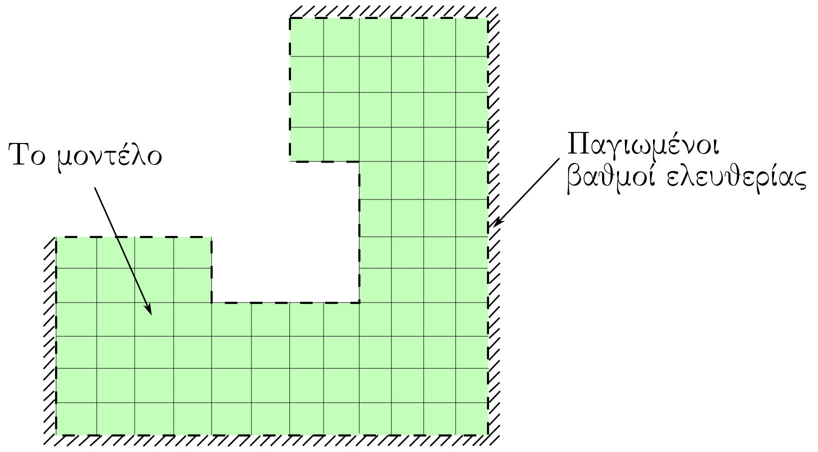
Σχήμα 4: Μια λωρίδα τεσσάρων στοιχείων της συμπληρωματικής περιοχής γύρω από το σύνορο

1. δημιουργία μοντέλου με τον αριθμό λωρίδων που έχουμε επιλέξει γύρω από το σύνορο
2. παγίωση του μοντέλου στην άλλη πλευρά
3. υπολογισμός προσέγγισης $\mathbf{D_\Gamma}$ με στατική συμπύκνωση στους εσωτερικούς βαθμούς ελευθερίας (Σχήμα 5)

Προσέγγιση μεγα-κλίμακας

Στην προσέγγιση μεγα-κλίμακας εφαρμόζουμε μερικές μετατοπίσεις στο σύνορο ώστε να δούμε την απόκριση ολόκληρου του φορέα. Η επιλογή του πεδίου αυτού των μετατοπίσεων είναι σημαντική για την ποιότητα της προσέγγισης που θα λάβουμε. Στη παρούσα μελέτη επιλέξαμε κινήσεις στερεού σώματος (μετατοπίσεις και στροφή) του συνόρου καθώς και παραμορφώσεις και στρεβλώσεις (Σχήμα 6).

Η μεθοδολογία για την προσεγγιση μεγα-κλίμακας είναι



Σχήμα 5: Το μοντέλο για την προσέγγιση μικρο-κλίμακας

1. Δημιουργία πίνακα **E**. Κάθε στήλη του πίνακα είναι και μια μορφή μετατόπισης του συνόρου ενώ οι σειρές είναι οι βαθμοί ελευθερίας του συνόρου
2. Επιβάλλουμε κάθε μορφή μετατόπισης (στηλη του **E**) στο σύνορο Γ του καθολικού φορέα και παίρνουμε τις δυνάμεις στο σύνορο.
3. Υπολογίζουμε τον πίνακα **P** από τη εξίσωση

$$\mathbf{E}^{\intercal}\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma,C}}\mathbf{EP} = \mathbf{I}$$

   Ο πίνακας $\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma,C}}\mathbf{E}$ είναι τα φορτία που βρήκαμε στο προηγούμενο βήμα.

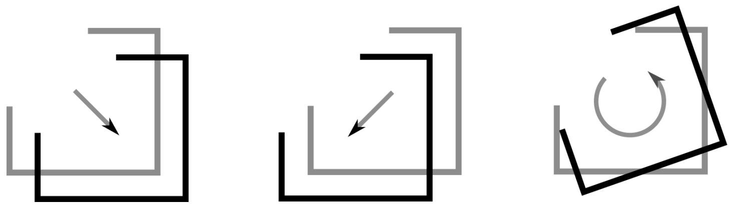
4. Υπολογίζουμε τον πίνακα **F** από την εξίσωση

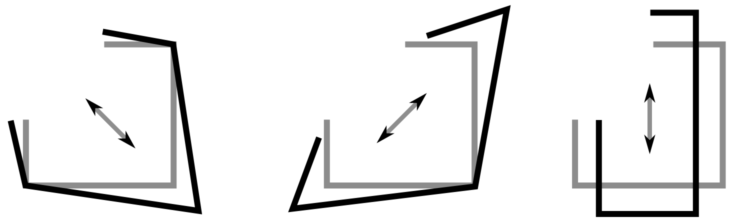
$$\mathbf{F} = \mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma,C}}\mathbf{EP}$$

Συνδιασμός των προσεγγίσεων

Από διάφορες μελέτες έχει διαπιστωθεί ότι οι παραπάνω προσεγγίσεις, η κάθε μια ξεχωριστά, δεν είναι επαρκείς. Φαίνεται λοιπόν ότι είναι απαραίτητο να λάβουμε υπόψιν μας τόσο την δυσκαμψία της εγγύτητας του συνόρου όσο και την συμπεριφορά ολόκληρου του φορέα σε μετακινήσεις του συνόρου. Προτείνεται λοιπόν ο συνδιασμός των παραπάνω προσεγγίσεων.

Η λογική μιας τέτοιας σκέψης βασίζεται στην αρχή του Saint-Venant, που είναι γνωστή και συχνά χρησιμοποιείται στη μηχανική και ιδιαίτερα στη θεωρία δοκού. Η αρχή αυτή δηλώνει ότι **οι εντάσεις που αναπτύσονται σε ένα**


(α΄) Μετατοπίσεις και στροφές

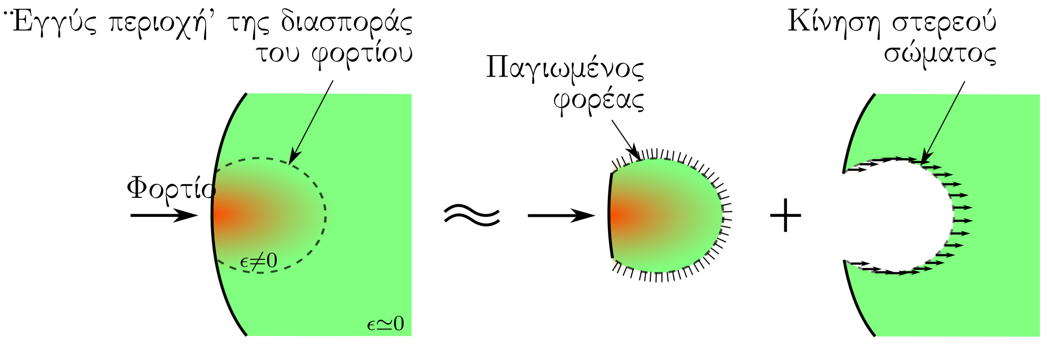

(β΄) Παραμορφώσεις και στρεβλώσεις

Σχήμα 6: Πεδίο μετατοπίσεων του συνόρου

φορέα από την εφαρμογή δυνάμεων σε μια μικρή περιοχή αυτο-ύ, είναι αμελητέου μεγέθους σε αποστάσεις που είναι απομακρυσμένες αρκετά από την περιοχή εφαρμογής των δυνάμεων. Με άλλα λόγια σε μεγάλες αποστάσεις σε σχέση με το πεδίο εφαρμογής τών δυνάμεων, η κίνηση του φορέα είναι σχεδόν στερεά (οι εντάσεις είναι αμελητέες). Αν μπορούσαμε να αναγνωρίσουμε αυτή την στερεά κίνηση και να την αφαιρέσουμε από το πεδίο των μετατοπίσεων του φορέα τότε οι εναπομείνουσες μετατοπίσεις θα ήταν σχεδόν μηδενικές. Οπότε η παγιώση του φορέα σε επαρκή απόσταση από το σημείο εφαρμογής των δυνάμεων δεν θα εισήγαγε σημαντικό σφάλμα στη λύση (Σχήμα 7).

Έχοντας υπολογίσει τις προσεγγίσεις μικρο-κλίμακας και μεγα-κλίμακας υπολογίζεται η συνδιασμένη προσέγγιση $\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}$ ώς εξής

$$\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}} = \mathbf{D}_{\mathbf{\Gamma}} - \mathbf{D}_{\mathbf{\Gamma}}\mathbf{U}\mathbf{C}\left[\mathbf{I} + \mathbf{V}\mathbf{D}_{\mathbf{\Gamma}}\mathbf{U}\mathbf{C}\right]^{-1}\mathbf{V}\mathbf{D}_{\mathbf{\Gamma}}$$



Σχήμα 7: Η αρχή του Saint Venant

όπου

$$\mathbf{U} = \begin{bmatrix} \mathbf{\Delta GF} & \mathbf{E} & \mathbf{E} \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{I} & 0 & 0 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & -\mathbf{F}^{\intercal}\mathbf{\Delta GF} \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} \mathbf{E}^{\intercal} \\ (\mathbf{\Delta GF})^{\intercal} \\ \mathbf{E}^{\intercal} \end{bmatrix}$$

και

$$\mathbf{\Delta GF} = \mathbf{EP} - \mathbf{D_\Gamma}^{-1}\mathbf{F}$$

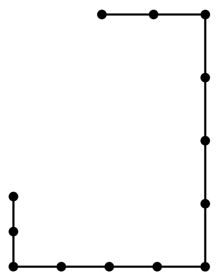
Υπολογισμός υπολοιπόμενων δυνάμεων

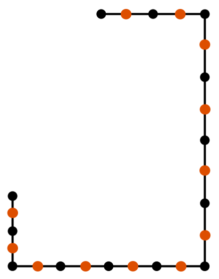
Έχοντας καθορίσει επιτυχώς το μητρώο **A** πρέπει να βρεθέι και η βέλτιστη λύση για το μητρώο **B**. Το πλεονέκτημα που θα μας δώσει αυτό το μητρώο έιναι ότι η καινούρια λύση του καθολικού προβλήματος δεν θα είναι πρακτικά μια διόρθωση της προηγούμενης λύσης του αλλά μια διόρθωση της λύσης του τοπικού προβλήματος,η οποία και βρίσκεται κοντύτερα στην τελική λύση. Προκύπτει λοιπόν ότι η καλύτερη λύση για το μητρώο **B** είναι

$$\mathbf{B} = \widetilde{\mathbf{S}}_{\mathbf{\Gamma}} - \mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}$$

Αυτό σημαίνει ότι το **B** προκύπτει από στατική συμπύκνωση των εσωτερικών βαθμών ελευθερίας του τοπικου μοντέλου, δηλαδή

$$\mathbf{B} = \mathbf{S}^{\mathbf{L}}_{\mathbf{\Gamma},\mathbf{I}}$$


(α΄) Σύνορο καθολικού μοντέλου


(β΄) Σύνορο τοπικού μοντέλου

Σχήμα 8: Μη συμβατά σύνορα

Μη συμβατά σύνορα

Είναι ιδιαίτερα σημαντικό να επεκταθεί η μέθοδος και για περιπτώσεις όπου το σύνορο του τοπικού μοντέλου δεν ταυτίζεται με το σύνορο του καθολικού μοντέλου. Έτσι θα υπάρχει η δυνατότητα το τοπικό μοντέλο να είναι πολύ διαφορετικό από το καθολικό δίνοντας μας μεγαλύτερη ελευθερία και περισσότερες δυνατότητες. Στη παρούσα διπλωματική παρουσιάζεται ένας τρόπος επίλυσης για περιπτώσεις όπου το τοπικό μοντέλο εμπλουτίζεται με περισσότερους κόμβους στο σύνορο (Σχήμα 8).

Ορμώμενοι από μεθόδους διακριτοποίησης φορέων δημιουργούμε ένα μητρώο $\mathbf{\Lambda}$ το οποίο συσχετίζει τις μετατοπίσεις των καινούριων κόμβων του τοπικού μοντέλου με τις μετατοπίσεις των κοινών κόμβων τοπικού και καθολικού μοντέλου. Έτσι για παράδειγμα για ένα κόμβο που βρίσκεται ανάμεσα σε δύο άλλους όπως φαίνεται στο Σχήμα 9 ισχύει

$$\mathbf{u}_C^L = \frac{1}{2}\mathbf{u}_A^G + \frac{1}{2}\mathbf{u}_B^G$$

και

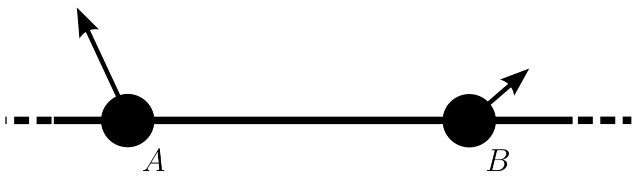
$$\mathbf{u}_A^L = \mathbf{u}_A^G$$
$$\mathbf{u}_B^L = \mathbf{u}_B^G$$

Εκφράζοντας τις παραπάνω σχέσεις σε μητρωική μορφή

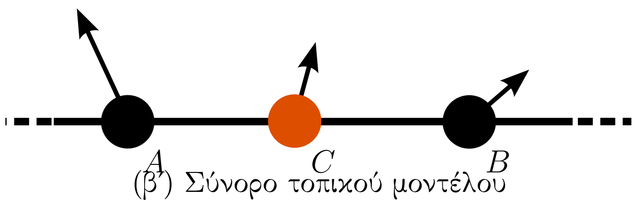
$$\begin{Bmatrix} \mathbf{u}_A \\ \mathbf{u}_B \\ \mathbf{u}_C \end{Bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_A \\ \mathbf{u}_B \end{Bmatrix}$$

Δημιουργείται το μητρώο $\mathbf{\Lambda}$ που συσχετίζει τις μετατοπίσεις του τοπικού συνόρου με αυτές του καθολικού.

$$\mathbf{u}_\Gamma^L = \mathbf{\Lambda}\mathbf{u}_\Gamma^G$$



(α΄) Σύνορο καθολικού μοντέλου



(β΄) Σύνορο τοπικού μοντέλου

Σχήμα 9: Συσχετισμός των νέων βαθμών ελευθερίας

Χρησιμοποιώντας το ανάστροφο μητρώο του $\mathbf{\Lambda}$ μπορούμε να κατανείμουμε τις δυνάμεις του τοπικού συνόρου στο καθολικό σύνορο

$$\mathbf{f}_\Gamma^G = \mathbf{\Lambda}^\intercal \mathbf{f}_\Gamma^L$$

Χρησιμοποιώντας το μητρώο $\mathbf{\Lambda}$ και το ανάστροφό του μπορούμε με μικρές αλλαγές στη μέθοδο να επιλύσουμε προβλήματα με μη συμβατα σύνορα.

Επίλογος

Παρουσιάστηκαν οι βασικές αρχές της μεθόδου που χρησιμοποιήθηκε σε αυτήν την διπλωματική για την συζευξη δυο μοντέλων. Ενός γραμμικού και ελαστικού ολόκληρου του φορέα και ενός τοπικού μη γραμικού για ένα μικρό μέρος του φορέα. Οι διάφορες παραλλαγές της μεθόδου παρουσιάστηκαν καθώς και ο τρόπος αντιμετώπισης μερικών περιπτώσεων μη συμβατότητας συνόρων. Παραδείγματα και αποτελέσματα τις μεθόδου παρουσιάζονται στα κεφάλαια 8 και 9 της παρούσας διπλωματικής.

1 Introduction

1.1 The Finite Element Method

The human mind, unable to comprehend the behaviour of its surroundings as a whole, has the tendency to divide all systems into their components or 'elements', whose behaviour can be understood. Then, by rebuilding the original system from these elements, the engineer or the scientist studies successfully its behaviour. This division can be done using a finite number of 'elements', called discrete problem, or using an infinite number of 'elements' (leading to differential equations or equivalent statements), called continuous problem.

Although discrete problems can be easily solved, especially with the advent of digital computers, continuous problems can only be solved exactly by mathematical manipulation, which is possible only for simple problems. Scientists and engineers have proposed various methods of discretization of the continuous problems. All these methods lead to an approximation, which approaches the exact solution as the number of discrete variables increases.

The problems that an engineer faces, such as problems of solid mechanics or fluid mechanics, are continuous problems. So the engineer approaches these problems by creating an analogy between real discrete elements and finite portions of the continuum domain. It is from the engineering 'direct analogy' view that the term 'finite element' was born. Throughout the years a standard methodology applicable to discrete systems was developed. The civil engineer, dealing with structures, first calculates force-displacement relationships for each element of the structure and then proceeds to assemble the whole by following a well-defined procedure of establishing local equilibrium at each 'node' or connecting point of the structure. The resulting equations can be solved for the unknown displacements. Similarly, the electrical or hydraulic engineer, dealing with a network of electrical components (resistors, capacitances, etc.) or hydraulic conduits, first establishes a relationship between currents (fluxes) and potentials for individual elements and then proceeds to assemble the system by ensuring continuity of flows.

The standard pattern that all these analyses follow is universally adaptable to discrete systems. "*The existence of a unified treatment of 'standard discrete problems' leads us to the first definition of the finite element process as a method of approximation to continuum problems such that*

1. *the continuum is divided into a finite number of parts (elements), the behaviour of which is specified by a finite number of parameters, and*

2. *the solution of the complete system as an assembly of its elements follows precisely the same rules as those applicable to standard discrete problems." [20]*

Through the work done in structural engineering during the nineteenth and twentieth centuries we can follow the development of the finite element method. We should mention here the work of Navier, Clebsch, Southwell and Cross before the second world war. Then with the larger use of matrices the finite elements took their current matrix form. The works of Duncan and Collar, Argyris, Kron and Turner played significant role in the formulation of the finite element method as it is today.

1.2 The Multiscale Methods

The Finite Element Method (FEM), assisted by the advances in computers, has been developed with great speed in both academia and industry the last century. Many various types of finite element methods have been introduced (GFEM, X-FEM, Meshfree methods etc.) and the research still goes on with tremendous results. In this plethora of methods and analysis there are problems and techniques that involve multiple scales.

Wherever large disparities in spatial and temporal scales are encountered the simulation efforts are dominated by multiple scales. Such disparities appear in all areas of modern science and engineering, for example, composite materials, porous media, turbulent transition in high Reynolds number flows, and so on. The direct numerical solution of multiple scale problems is difficult even with the advent of supercomputers. The major difficulty of direct solutions is the size of the computation. The situation can be relieved to some degree by parallel computing. However, the size of the discrete problem is not reduced. Therefore, it would be desirable to develop a method that captures the small-scale effect on the large scales, but does not require solving all the small scale features. Single-scale models, usually at a macro scale, make use of constitutive equations which should reflect the behaviour of the underlying finer scales, but in many cases the microstructure may not be scalable and may be different at each structural level [19]. These constitutive equations are generally of a phenomenological type.

An alternative to the use of constitutive equations at a single (macro) scale is provided by multiscale modelling, in which the relevant physics is explicitly captured on multiple spatial and temporal scales. There are various techniques of multiscale analysis. Most common are self-consistent methods, asymptotic analysis and homogenization techniques [6].

In this thesis we will investigate the case where a complex structure, that mostly evolves in a linear elastic way, exhibits confined plasticity (or other non-linear phenomena) in a small critical region. This is a common case in many engineering problems, especially in the aircraft industry. This problem occurs in two scales. The one is the large linear model of the structure and the other is the small region that the non-linearity occurs. Similar problems are those where, instead of a small non-linearity, we have either strange geometry (cracks or holes) or where we want to use different types of FEM [9]. Also situations where an existing model (e.g. a building) has been modified in a small region (local failure or reinforcement) are identical problems.

1.3 Outline of the diploma thesis

The present diploma thesis focuses on intermediate ways to analyze complex structures that contain small nonlinear areas, when full nonlinear computations are too expensive. It has been based mainly on the work of Gendre et al. [8] [7] and proposes algorithms for the implementation of their methods and a simple extension on non-conforming meshes. The remainder of the paper is structured as follows.

- In the **second chapter** the fundamentals of the non-linear solid mechanics are presented. To be more specific, the von Mises plasticity with no hardening is presented and the forward and backward Euler schemes. A chapter with elasticity theory was avoided as it is regarded a well-known subject of most engineers. This chapter, as well as the third one, contain the essential theory for the creation of the basic non-linear finite element software that was used for this thesis.

- The **third chapter** contains various non-linear analysis methods. In the same logic, algorithms for the analysis of linear problems are considered known [18] and someone can append in the bibliography for further reading.

- In the **fourth chapter** the iterative method, on which the whole thesis is based, is presented. It is the method that connects the local model, which describes the small non-linear area, with the global model. The two variations of this method are presented, the displacement and the mixed variation.

- The **fifth chapter** describes the procedure for the creation of a Schur complement approximation that is used in the mixed variation method. The procedure involves both short-scale techniques and long-scale techniques that are combined for the creation of the approximation.

- The **sixth chapter** offers a solution for the handling of some cases of non-conforming meshes. Inspired from domain decomposition techniques a matrix that connects the different interfaces is created and later used in the method.

- The **seventh chapter** describes the whole procedure of the creation of the software that was used for the current thesis. It describes how all the theory was turned into algorithms. It also includes segments of the matlab code that was created. Such segments are found in various places in this thesis, whenever their appearance thought to be supporting for the comprehension of the text.

- **Chapters eight and nine** contain the two examples of this thesis. The global domain is the same for both examples, but the local domain is different. The first example, in which the local model has the same mesh with the global, focuses on the comparison of the methods. On the other hand, the second example, in which the local model has different mesh than the global, focuses on the use of non-conforming meshes.

- Finally, **chapter ten** phrases the conclusions of the whole study and ideas for future research.

2 Non Linear Material in Finite Element Method

2.1 Introduction

There are many types of non linearities in FEM. For the current thesis a material non linearity, with a von Mises yield criterion was chosen, based on [5] [13]. The material that was chosen was a bilinear elastic fully plastic material. Of course there are many other criteria that can be applied easily such as the Mohr Coulomb as well as many non linear laws to choose from. Nevertheless, the purpose of the thesis was not to examine differences between the different kind of non linearities, but to examine the connection of the global linear model and the local non linear. This is the reason why a simple non linear FEM was chosen for the algorithm.

The elasto-plastic material, contrary to the linear elastic, having reached equilibrium at point A (Figure (2.1)) on the effective stress/strain curve, will continue to flow plastically to point B (remaining on the effective stress/strain curve) and then continue to point C on the curve or unload elastically to D. Clearly the two paths have different stiffness, but in the absence of prior knowledge on load reversals, it will generally be assumed that plastic flow will continue and that the tangent stiffness relates to BC. However, even for monotonically increasing loads, certain areas of the structure can 'unload'. In such circumstances, it is usually left to the iterative correction procedure to discover those areas that are 'unloading'.

In general, there are three separate roles for the plasticity algorithms of a finite element code. Those roles are:

1. the formation of the standard tangent modular matrix to be used in the incremental tangent stiffness matrix or to be used with the integration of the stress/stain laws

2. the formation of a 'consistent' tangent modular matrix to be used with

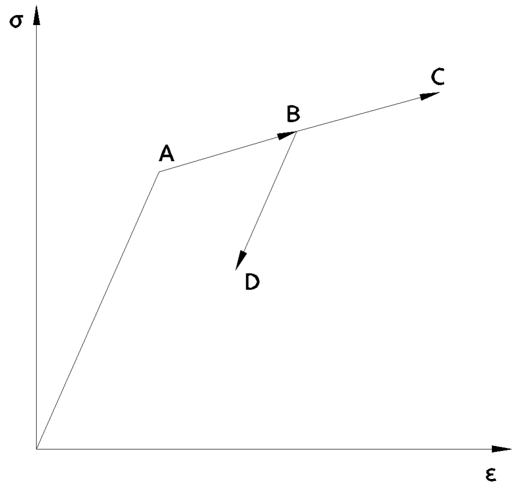


Figure 2.1: One dimentional stress-strain relationship

Newton-Raphson iterations

3. the integration of the stress/strain laws to update the stresses

With material non-linearity changing the modular matrix, the structural tangent stiffness matrix takes the form

$$\mathbf{K_t} = \int \mathbf{B}^{\intercal}\mathbf{C_t}\mathbf{B}dV \tag{2.1}$$

where $\mathbf{C_t}$ is the standard tangential modular matrix, which is given by

$$\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\epsilon}} = \mathbf{C_t} \tag{2.2}$$

If certain forms of stress updating are adopted, it is possible to derive a 'consistent' tangent modular matrix, $\mathbf{C_t}$, that is consistent with the numerical technique used for the stress updating. In such cases, for example when the Newton-Raphson algorithm is used for the equilibrium iterations, the consistent tangent modular matrix leads to a significantly faster convergence rate than the 'standard' tangential modular matrix.

2.2 The standard elasto-plastic modular matrix for an elastic perfectly plastic von Mises material under plane stress

In some senses plane stress is one of the more difficult stress state. However, plane stress will be used to introduce plasticity calculations simply because it involves fewer components. The prime aim is to develop the general form of the matrix, vector and tensor equations which will also apply to more general stress states.

We will start with the simple plane-stress version of the von Mises yield function

$$f = (\sigma_x^2 + \sigma_y^2 - \sigma_x\sigma_y + 3\tau_{xy}^2)^{\frac{1}{2}} - \sigma_0 = \sigma_e - \sigma_0 \tag{2.3}$$

where σ_e is the effective stress and σ_0 the yield stress. In conjunction with (2.3), the Prandlt-Reuss flow rules are

$$\dot{\epsilon}_p = \dot{\lambda}\left(\frac{\partial f}{\partial \boldsymbol{\sigma}}\right) = \dot{\lambda}\mathbf{a} = \begin{pmatrix} \dot{\epsilon_{px}} \\ \dot{\epsilon_{py}} \\ \dot{\epsilon_{pxy}} \end{pmatrix} = \frac{\dot{\lambda}}{2\sigma_e}\begin{pmatrix} 2\sigma_x - \sigma_y \\ 2\sigma_y - \sigma_x \\ 6\tau_{xy} \end{pmatrix} \tag{2.4}$$

the vector $\mathbf{a}$ is normal to the yield surface and $\dot{\lambda}$ is a positive constant usually referred to as 'plastic strain-rate multiplier'. (Note that with the present notation, $\partial f/\partial\boldsymbol{\sigma}$ is a column vector.) In equation (2.4) and generally during this chapter, the 'rate' form is denoted by a dot. However, we are not considering dynamic effects. As a result we have a 'pseudo-time' and indeed the dotted quantities can be simply considered as small changes are usually designated via δs. In addition to (2.4), the stress changes are related to strain changes via

$$\dot{\boldsymbol{\sigma}} = \begin{pmatrix} \dot{\sigma_x} \\ \dot{\sigma_y} \\ \dot{\sigma_{xy}} \end{pmatrix} = \mathbf{C}\left(\begin{pmatrix} \dot{\epsilon_x} \\ \dot{\epsilon_y} \\ \dot{\epsilon_{xy}} \end{pmatrix} - \begin{pmatrix} \dot{\epsilon_{px}} \\ \dot{\epsilon_{py}} \\ \dot{\epsilon_{pxy}} \end{pmatrix}\right) = \mathbf{C}(\dot{\boldsymbol{\epsilon}} - \dot{\boldsymbol{\epsilon}_\mathbf{p}}) = \mathbf{C}(\dot{\boldsymbol{\epsilon}} - \dot{\lambda}\mathbf{a}) \tag{2.5}$$

where, assuming isotropic elasticity,

$$\mathbf{C} = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \tag{2.6}$$

Equation (2.5) relates the small changes in stress (or more strictly stress rates) to the small changes in elastic strain (or more strictly elastic strain rates), $\dot{\boldsymbol{\epsilon}_\mathbf{e}} = \dot{\boldsymbol{\epsilon}} - \dot{\boldsymbol{\epsilon}_\mathbf{p}}$.

A negative 'plastic strain-rate multiplier', $\dot{\lambda}$, would imply plastic unloading from the yield surface. The latter cannot occur and, consequently, any negative, $\dot{\lambda}s$ should be replaced by zero so that elastic unloading occurs.

For plastic flow to occur, the stresses must remain on the yield surface and hence

$$\dot{f} = \frac{\partial f^{\mathsf{T}}}{\partial \boldsymbol{\sigma}} \dot{\boldsymbol{\sigma}} = \mathbf{a}^{\mathsf{T}} \dot{\boldsymbol{\sigma}} = 0 \tag{2.7}$$

The situation described by (2.7) shows that, for plastic flow, the stress changes, $\dot{\boldsymbol{\sigma}}$, are instantaneously moving tangentially to the surface, with $\dot{\boldsymbol{\sigma}}$ being orthogonal to the vector $\mathbf{a}$. Hence $\mathbf{a}$ is normal to the surface and the flow rules (equation (2.4)) invoke normality.

In order to find the plastic strain-rate multiplier, λ, equation (2.5) is premultiplied by the flow vector $\mathbf{a}^{\mathsf{T}}$ and, using equation (2.7)

$$\dot{\lambda} = \frac{\mathbf{a}^{\mathsf{T}} \mathbf{C} \dot{\boldsymbol{\epsilon}}}{\mathbf{a}^{\mathsf{T}} \mathbf{C} \mathbf{a}} \tag{2.8}$$

Consequently, substitution into equation (2.5) gives

$$\dot{\boldsymbol{\sigma}} = \mathbf{C_t} \dot{\boldsymbol{\epsilon}} = \mathbf{C} \left(I - \frac{\mathbf{a}\mathbf{a}^{\mathsf{T}} \mathbf{C}}{\mathbf{a}^{\mathsf{T}} \mathbf{C} \mathbf{a}} \right) \dot{\boldsymbol{\epsilon}} \tag{2.9}$$

where $\mathbf{C_t}$ is the tangential modular matrix which is not only a function of E and ν but also, via a, a function of the current stresses, $\boldsymbol{\sigma}$. This matrix can now be used in finite element expressions to form the element and hence the structure tangent stiffness matrix.

2.3 Von Mises plasticity in three dimensions

For the general three-dimensional case, the von MIses yield criterion is

$$\begin{aligned}
f &= \sigma_e - \sigma_0 = \sqrt{3} J_2^{\frac{1}{2}} - \sigma_0 \\
&= \frac{1}{\sqrt{2}} [(\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_z - \sigma_x)^2 + 6(\tau_{xy}^2 + \tau_{yz}^2 + \tau_{zx}^2)]^{\frac{1}{2}} - \sigma_0 \\
&= \sqrt{3} [\frac{1}{2}(s_x^2 + s_y^2 + s_z^2) + \tau_{xy}^2 + \tau_{yz}^2 + \tau_{zx}^2]^{\frac{1}{2}} - \sigma_0 \\
&= \sqrt{\frac{3}{2}} (\mathbf{s}^{\mathsf{T}} \mathbf{L} \mathbf{s})^{\frac{1}{2}} - \sigma_0
\end{aligned} \tag{2.10}$$

where

$$\mathbf{L} = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 2 & & \\ & & & & 2 & \\ & & & & & 2 \end{bmatrix} \tag{2.11}$$

and

$$\mathbf{s}^{\mathsf{T}} = \{s_x, s_y, s_z, \tau_{xy}, \tau_{yz}, \tau_{zx}\} \tag{2.12}$$

are the deviatoric stresses.

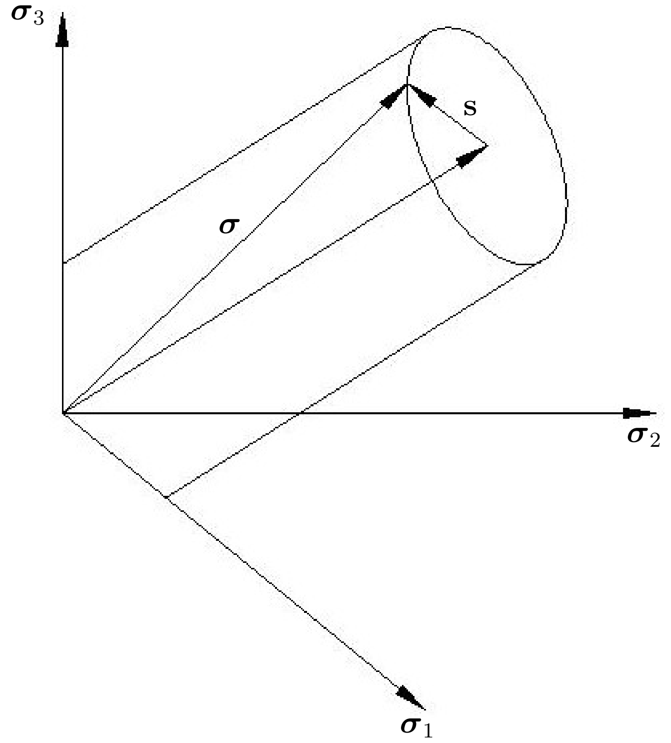


Figure 2.2: Von Mises yield criterion in three-dimensional principal stresses space

The three-dimensional von Mises yield criterion is plotted in principal stresses space in Figure (2.2) where the stress vector $\boldsymbol{\sigma}$ is decomposed into a volumetric component (along the axis $i^{\mathsf{T}} = (1, 1, 1)$) and a devatoric component, $\mathbf{s}$. From (2.10), the radius of the von Mises cylinder is clearly $\sqrt{\frac{2}{3}}\sigma_0$.

For three-dimensional plasticity, the equivalent plastic strain rate is given by

$$\dot{\epsilon}_{ps} = \sqrt{\frac{2}{3}} \left[\dot{\epsilon}_{px}^2 + \dot{\epsilon}_{py}^2 + \dot{\epsilon}_{pz}^2 + \frac{1}{2} (\dot{\gamma}_{xy}^2 + \dot{\gamma}_{yz}^2 + \dot{\gamma}_{zx}^2) \right]^{\frac{1}{2}} \tag{2.13}$$

The elastic stresses and strains are connected by

$$\begin{pmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{pmatrix} = [C] \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{pmatrix} \text{ or } \boldsymbol{\sigma} = [C]\boldsymbol{\epsilon} \tag{2.14}$$

where

$$[C] = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} (1-\nu) & \nu & \nu & & & \\ \nu & (1-\nu) & \nu & & & \\ \nu & \nu & (1-\nu) & & & \\ & & & \frac{1}{2}(1-2\nu) & & \\ & & & & \frac{1}{2}(1-2\nu) & \\ & & & & & \frac{1}{2}(1-2\nu) \end{bmatrix} \tag{2.15}$$

Differentiating equation (2.10) gives

$$\begin{aligned} \mathbf{a}^{\intercal} &= \frac{\partial f^{\intercal}}{\partial \boldsymbol{\sigma}} \\ &= \frac{1}{2\sigma_e} \{(2\sigma_x - \sigma_y - \sigma_z), (2\sigma_y - \sigma_x - \sigma_z), (2\sigma_z - \sigma_x - \sigma_y), 6\tau_{xy}, 6\tau_{yz}, 6\tau_{zx}\} \\ &= \frac{3}{2\sigma_e} \{s_x, s_y, s_z, \tau_{xy}, \tau_{yz}, \tau_{zx}\} = \frac{3}{2\sigma_e} (\mathbf{Ls})^{\intercal} = \frac{\partial f^{\intercal}}{\partial \mathbf{s}} \end{aligned} \tag{2.16}$$

As in (2.4), $\dot{\boldsymbol{\epsilon}_{\mathbf{p}}} = \dot{\lambda}\mathbf{a}$ so that in (2.13)

$$\dot{\epsilon}_{ps} = \sqrt{\frac{2}{3}} \dot{\lambda} (\mathbf{a}^{\intercal}\mathbf{L}^{-1}\mathbf{a})^{\frac{1}{2}} = \sqrt{\frac{2}{3}} \frac{\dot{\lambda}}{\sigma_e} (\mathbf{s}^{\intercal}\mathbf{Ls})^{\frac{1}{2}} \tag{2.17}$$

With these definitions of $\boldsymbol{\sigma}$, $\boldsymbol{\epsilon}$, $\mathbf{C}$, and a, an identical formulation of Section 2.2 produces equations (2.8) for $\dot{\lambda}$ and (2.9) for $\mathbf{C_t}$

2.4 Integrating the rate equations

The solution procedure (incremental predictor/corrector approaches), as it is based on the incremental (or rate) nature of the flow rules, inevitably leads to some error. This error will not relate to a lack of equilibrium, but rather will be caused by errors in the integration of the flow rules and their relation to the complete incremental/iterative solution procedure. Consequently, the adopted procedure for updating the stresses and strains affects the error. The recommended procedure is the one 'using the incremental strains':

1. Compute the iterative displacements, $\delta \mathbf{p}$, using, for example, $\delta \mathbf{p} = -\mathbf{K_t^{-1}} g$
2. Update the incremental displacements (from the last converged equilibrium state) using $\Delta \mathbf{p}_n = \Delta \mathbf{p}_0 + \delta \mathbf{p}$, where $\Delta \mathbf{p}_0$ are the incremental displacements at the end of the last iteration
3. Compute the incremental strains, $\Delta \boldsymbol{\epsilon}$, from the incremental displacements, $\Delta \mathbf{p}$, using $\Delta \boldsymbol{\epsilon} = fn(\Delta \mathbf{p})$
4. Compute the incremental stresses preferably by integrating the rate equations
5. Update the stresses using, $\boldsymbol{\sigma}_n = \boldsymbol{\sigma}_0 + \Delta \boldsymbol{\sigma}$ where $\boldsymbol{\sigma}_0$ are the old stresses at the end of the last increment

Using this procedure, the incremental stress is simply re-computed from the new incremental strain. The main advantages of this algorithm are gained because the stresses are always updated from the stresses at the end of the last increment. These stresses are in equilibrium.

If the stress and strains increments were very small, we could effectively proceed by applying the previous tangential formula (equation (2.9)) with terms like $\dot{\boldsymbol{\epsilon}}$ being replaced by terms like $\delta \boldsymbol{\epsilon}$ and use the strain updating scheme of the above procedure. In this case, instead of using equation (2.9), it would be more efficient to separately use (2.8) to compute $\dot{\lambda}$ and hence knowing $\mathbf{a} = \frac{\partial f}{\partial \sigma}$, to compute $\dot{\boldsymbol{\sigma}}$ from the general form in (2.5). However, the strain and subsequent stress changes will not be infinitesimally small and, as a consequence, errors would accumulate just as they would under other schemes such as the pure 'incremental' or 'forward-Euler scheme' and we cannot replace terms like $\dot{\boldsymbol{\epsilon}}$ with terms like $\Delta \boldsymbol{\epsilon}$. For the von Mises yield criterion, we can however add a higher-order term and replace (2.7) by

$$\Delta f = \mathbf{a}^{\mathsf{T}} \Delta \boldsymbol{\sigma} + \frac{1}{2} \Delta \boldsymbol{\sigma}^{\mathsf{T}} \frac{\partial \mathbf{a}}{\partial \boldsymbol{\sigma}} \Delta \boldsymbol{\sigma} \tag{2.18}$$

where differentiation of (2.16) gives

$$\frac{\partial \mathbf{a}}{\partial \boldsymbol{\sigma}} = \frac{1}{2\sigma_e}\begin{bmatrix} 2 & -1 & -1 & & & \\ -1 & 2 & -1 & & & \\ -1 & -1 & 2 & & & \\ & & & 6 & & \\ & & & & 6 & \\ & & & & & 6 \end{bmatrix} - \frac{1}{\sigma_e}\mathbf{a}\mathbf{a}^{\mathsf{T}} = \frac{1}{2\sigma_e}\mathbf{A} - \frac{1}{\sigma_e}\mathbf{a}\mathbf{a}^{\mathsf{T}} \qquad (2.19)$$

If we simply calculate $\mathbf{a} = \frac{\partial f}{\partial \boldsymbol{\sigma}}$ at the beginning of the increment and use equation (2.8) to compute $\Delta\lambda$, we adopt 'forward-Euler scheme' which is bound to lead to stresses that lie outside the yield surface at the end of the increment. Unless steps are taken to return the stresses to the yield surface or in some other way to ensure that the stresses remain at least very close to surface, errors are bound to accumulate and the computed collapse load will generally be overpredicted. There are three procedures which can be used, either individually or in combination to overcome this problem. They are

1. Add a return to the yield surface to the 'forward-Euler' scheme
2. Use sub-increments
3. Use same form of backward or mid-point Euler scheme

2.4.1 'Forward-Euler scheme and return to the yield surface

The 'forward-Euler' scheme has two steps. Firstly, the intersect point with the yield surface is found and then the tangent modular matrix is computed and the new stresses are calculated. To find the intersection, we require

$$f(\boldsymbol{\sigma}_X + \alpha\Delta\boldsymbol{\sigma}_e) = 0 \qquad (2.20)$$

where the original stresses, $\boldsymbol{\sigma}_X$ are such that

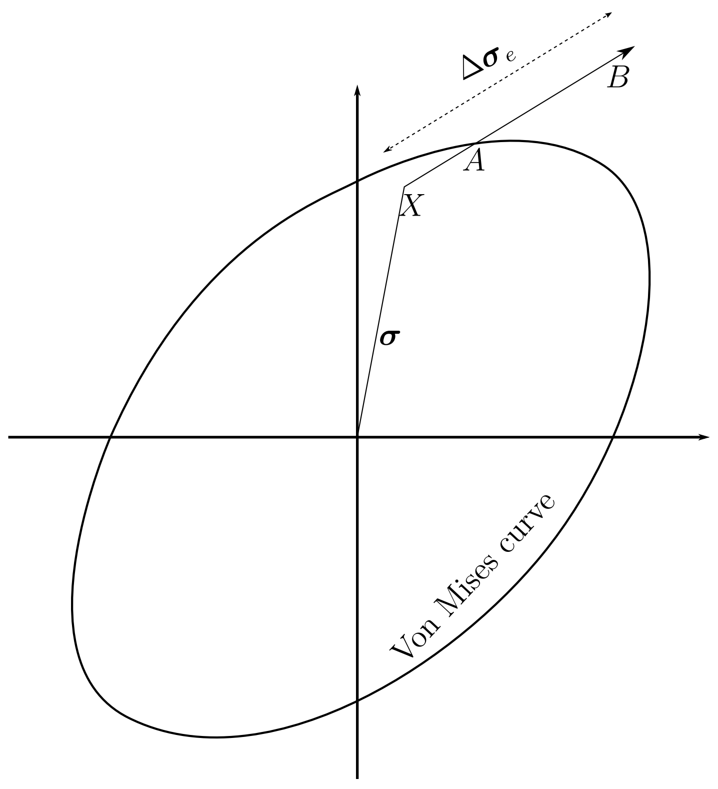
$$f(\boldsymbol{\sigma}_X) = f_X < 0 \qquad (2.21)$$

while, with $a = 1$, the elastic stresses $\boldsymbol{\sigma}_X + \Delta\boldsymbol{\sigma}_e$ give

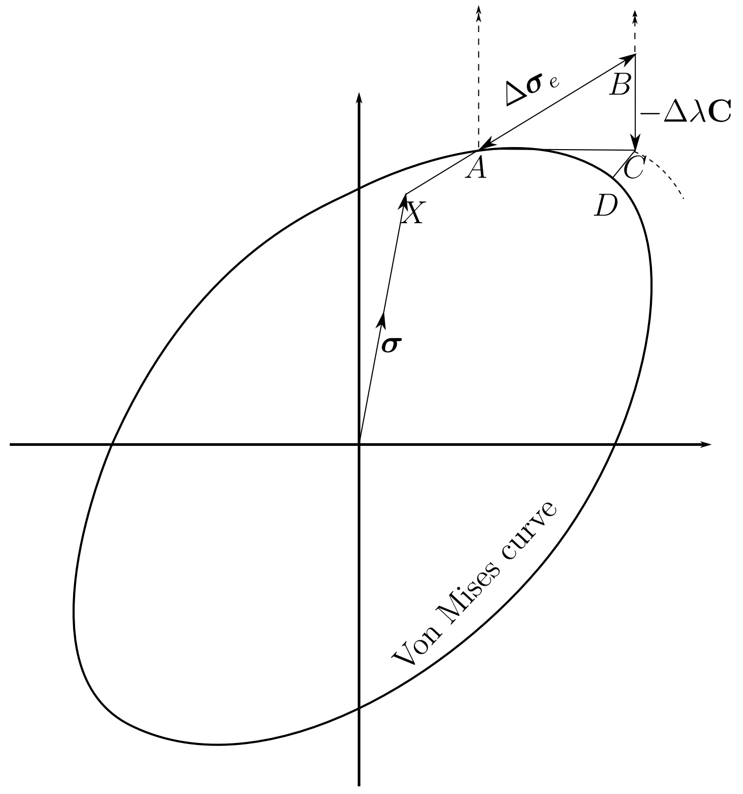
$$f(\boldsymbol{\sigma}_B) = f(\boldsymbol{\sigma}_X + \Delta\boldsymbol{\sigma}_e) > 0 \qquad (2.22)$$

For some yield surfaces, this problem can be solved exactly. For example, with the von Mises yield function, we can use $\mathbf{A}$ matrix in (2.19) to re-express the yield function (2.10) in squared form as

$$f_2 = \sigma_e^2 + \sigma_0^2 = \frac{1}{2}\boldsymbol{\sigma}^{\mathsf{T}}\mathbf{A}\boldsymbol{\sigma} - \sigma_0^2 = 0 \qquad (2.23)$$



(a) Locating the intersection point A



(b) Moving tangentialy from A to C (and correcting to D)

Figure 2.3: The forward-Euler procedure

Substituting the stresses, $\boldsymbol{\sigma}_X + a\Delta\boldsymbol{\sigma}_e$ into (2.23) gives

$$f_2 = \alpha^2\sigma_e(\Delta\boldsymbol{\sigma}_e)^2 + \alpha\Delta\boldsymbol{\sigma}_e^{\intercal}\mathbf{A}\boldsymbol{\sigma}_X + \sigma_e(\sigma_X)^2 - \sigma_0^2 = 0 \tag{2.24}$$

where the σ_e terms are simply the 'equivalent stress' terms of (2.10). We require the positive root of (2.24).

Alternatively, for a general yield function we can use a truncated Taylor series with a as the only variable to set up an iterative scheme. Such a scheme might start with an initial estimate

$$\alpha_0 = \frac{-f_X}{f_B - f_X} \tag{2.25}$$

and then use the truncated Taylor series

$$f_n = f_0 + \frac{\partial f \partial\boldsymbol{\sigma}}{\partial\boldsymbol{\sigma}\partial\alpha}\delta\alpha = f_0 + \mathbf{a}^{\intercal}\Delta\boldsymbol{\sigma}_e\delta\alpha = 0 \tag{2.26}$$

to give a first change in α, $\delta\alpha_0$. In applying (2.26), the 'old' yield function value, f_0, would for the iteration be computed from the stresses $\boldsymbol{\sigma} = \boldsymbol{\sigma}_{\mathbf{X}} + \alpha_0\Delta\boldsymbol{\sigma}_e$ with f_0 being computed from the same stresses. The scalar α would then be updated using $\alpha_1 = \alpha_0 + \delta\alpha_0$ while a second iteration would involve

$$\delta\alpha_1 = \frac{-1}{\mathbf{a}^{\intercal}\Delta\boldsymbol{\sigma}_e} f_1 \tag{2.27}$$

where a and f_1 would be computed at α_1. Having computed the intersection point $\boldsymbol{\sigma}_X + a\Delta\boldsymbol{\sigma}_e$, the remaining portion of the strain increment, which is $(1-\alpha)\Delta\boldsymbol{\sigma}$, can be treated in an elasto-plastic manner.

Matlab Code 2.1: Function factor to yield surface

```matlab
function a=factorToYieldSurface(self,currentStresses,stressesIncrement)
            fx=self.damageFunction(currentStresses);
            fb=self.damageFunction(currentStresses+stressesIncrement);
            a0=-fx/(fb-fx);
            f0=self.damageFunction(currentStresses+a0*stressesIncrement);
            ai=a0;
            fi=f0;
            while abs(fi)>10e-5
                atangenti=NonLinearMaterial3D.verticalVectorToYieldSurface↩
                    (currentStresses+ai*stressesIncrement);
                dai=-1/(atangenti'*stressesIncrement)*fi;
                ai=ai+dai;
                fi=self.damageFunction(currentStresses+ai*↩
                    stressesIncrement);
            end
            a=ai;
        end
```

The tangent modular matrix $\mathbf{C_t}$ is then computed as in equation (2.9)

$$\mathbf{C_{tA}} = \mathbf{C}\left(I - \frac{\mathbf{a}\mathbf{a}^\intercal\mathbf{C}}{\mathbf{a}^\intercal\mathbf{C}\mathbf{a}}\right)\Bigg|_A \tag{2.28}$$

Hence the forward-Euler scheme leads to

$$\boldsymbol{\sigma}_C = \boldsymbol{\sigma}_A + \mathbf{C_{tA}}\Delta\boldsymbol{\epsilon} \tag{2.29}$$

Matlab Code 2.2: Function stress increment from yield surface

```
function stressIncrement=stressIncrementFromYieldSurface(self,↩
    currentStresses,strainIncrement,steps)
            stressIncrement=0;
            for i=1:steps
                Ct=self.getTangentModularMatrix(currentStresses+↩
                    stressIncrement);
                stressIncrement=stressIncrement+Ct*(strainIncrement./steps↩
                    );
            end
        end
```

Alternatively, $\Delta\lambda$ could be computed from (2.8) but with Δ's instead of dots so

$$\Delta\lambda = \frac{\mathbf{a}^\intercal\mathbf{C}\Delta\boldsymbol{\epsilon}}{\mathbf{a}^\intercal\mathbf{C}\mathbf{a}} \tag{2.30}$$

and then

$$\Delta\boldsymbol{\sigma}_C = \mathbf{C}\Delta\boldsymbol{\epsilon} - \Delta\lambda\mathbf{C}\mathbf{a} \tag{2.31}$$

For the return to the yield function we require

$$\boldsymbol{\sigma}_D = \boldsymbol{\sigma}_C - \delta\lambda_C\mathbf{C}\mathbf{a}_C \tag{2.32}$$

where

$$\delta\lambda_C = \frac{f}{\mathbf{a}^\intercal\mathbf{C}\mathbf{a} + \mathbf{A}}\Bigg|_C \tag{2.33}$$

and $\boldsymbol{\sigma}_D$ are the new stresses closer to the yield surface, while $\boldsymbol{\sigma}_C$ are the old stresses with damage function greater than zero.

Matlab Code 2.3: Function correction to yield surface

```
function stressDecrement=correctionToYieldSurface(self,currentStresses)

            fc=self.damageFunction(currentStresses);
            a=NonLinearMaterial3D.verticalVectorToYieldSurface(↩
                currentStresses);
            dl=fc/(a'*self.C6*a);
            stressDecrement=dl*self.C6*a;
        end
```

Matlab Code 2.4: Function forward Euler with return

```
function [newStresses isNonlinear]=forwardEulerWithReturn(self,↩
    currentStresses,strainIncrement)
            stressesIncrement=self.C6*strainIncrement;
            a=self.factorToYieldSurface(currentStresses,stressesIncrement)↩
                ;
            if a<1
                isNonlinear=true;
                intersectStresses=currentStresses+a*stressesIncrement;
                remainingStrains=(1-a)*strainIncrement;
                newStresses=intersectStresses+self.↩
                    stressIncrementFromYieldSurface(intersectStresses,↩
                    remainingStrains,1);
                while self.damageFunction(newStresses)>self.↩
                    damageMagnitude
                    newStresses=newStresses-self.correctionToYieldSurface(↩
                        newStresses);
                end
            else
                isNonlinear=false;
                newStresses=currentStresses+(self.C6*strainIncrement);
            end

        end
```

2.4.2 Sub-increments

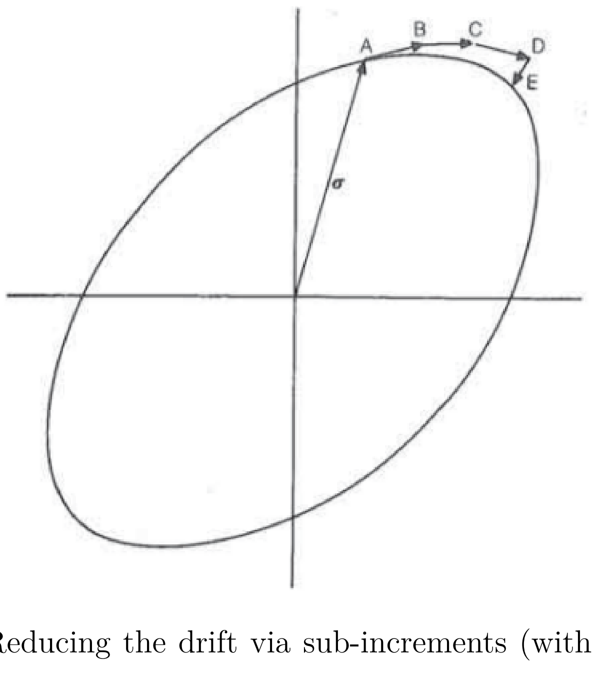


Figure 2.4: Reducing the drift via sub-increments (with later correction to E)

The use of sub-incrementation can be significantly reduce the errors that are introduced by the forward-Euler tangential scheme. In this technique the incremental strain $\Delta\boldsymbol{\epsilon}$ is divided into m sub-steps and the standard forward-Euler procedure is applied at each step. Even with sub-incrementation it is probably wise to introduce a 'correction' either at the end of each sub-step or at the end of the increment. The sub-incrementation will not be discussed further as it was not used in the thesis.

2.4.3 Backward-Euler scheme

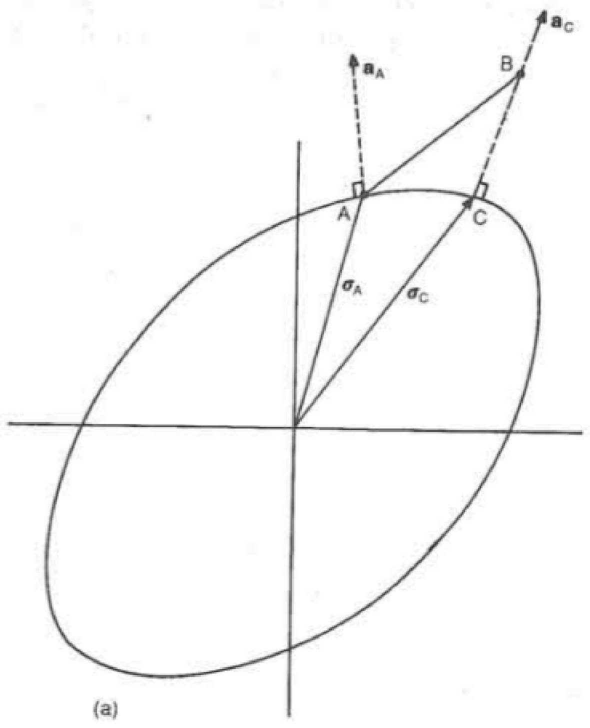


Figure 2.5: Backward-Euler return

The 'backward-Euler' scheme is the one that is mainly used with Newton-Raphson techniques as it has some advantages (simple formulation and quadratic convergence) over the 'forward-Euler' scheme. This scheme has also a specific plane-stress formulation which is more efficient at such problems. The backward-Euler return is based on the equation

$$\boldsymbol{\sigma}_C = \boldsymbol{\sigma}_B - \Delta\lambda \mathbf{C}\mathbf{a}_C \tag{2.34}$$

and a starting estimate for $\boldsymbol{\sigma}_C$ is needed. Generally the starting estimate will not satisfy the yield function and further iteration will be required because the normal at the trial position B (Figure (2.5)) will not general equal the final normal. In order to derive such an iterative loop, a vector **r**, can be set up to represent the difference between the current stresses and the backward-Euler stresses,

$$\mathbf{r} = \boldsymbol{\sigma} - (\boldsymbol{\sigma}_B - \Delta\lambda \mathbf{C}\mathbf{a}_C) \tag{2.35}$$

and iterations are introduced to reduce **r** to (almost) zero while the final stresses would satisfy the yield criterion, $f = 0$.

With the trial elastic stresses, $\boldsymbol{\sigma}_B$ being kept fixed, a truncated Taylor expansion can be applied to equation (2.35) so as to produce a new residual, $\mathbf{r}_n$, where

$$\mathbf{r}_n = \mathbf{r}_0 + \dot{\boldsymbol{\sigma}} + \dot{\lambda}\mathbf{C}\mathbf{a} + \Delta\lambda\mathbf{C}\frac{\partial \mathbf{a}}{\partial \boldsymbol{\sigma}}\dot{\boldsymbol{\sigma}} \tag{2.36}$$

$\dot{\boldsymbol{\sigma}}$ is the change in $\boldsymbol{\sigma}$ and $\dot{\lambda}$ is the change in $\Delta\lambda$. Setting $\mathbf{r}_n$ to zero gives

$$\dot{\boldsymbol{\sigma}} = -\left(\mathbf{I} + \Delta\lambda\mathbf{C}\frac{\partial \mathbf{a}}{\partial \boldsymbol{\sigma}}\right)^{-1}\left(\mathbf{r}_0 + \dot{\lambda}\mathbf{C}\mathbf{a}\right) = -\mathbf{Q}^{-1}\mathbf{r}_0 - \dot{\lambda}\mathbf{Q}^{-1}\mathbf{C}\mathbf{a} \tag{2.37}$$

Also, a truncated Taylor series on the yield function gives

$$f_{C_n} = f_{C_0} + \frac{\partial f^{\intercal}}{\partial \boldsymbol{\sigma}}\dot{\boldsymbol{\sigma}} = f_{C_0} + \mathbf{a}_C^{\intercal}\dot{\boldsymbol{\sigma}} = 0 \tag{2.38}$$

so that (dropping the subscript C)

$$\dot{\lambda} = \frac{f_0 - \mathbf{a}^{\intercal}\mathbf{Q}^{-1}\mathbf{r}_0}{\mathbf{a}^{\intercal}\mathbf{Q}^{-1}\mathbf{C}\mathbf{a}} \tag{2.39}$$

Matlab Code 2.5: Function backward-Euler return

```
function [newStresses isNonlinear]=backwardEulerReturn(self,↩
    currentStresses,strainIncrement)
            C=self.C6;

            stressesIncrement=C*strainIncrement;
            newStressesB=currentStresses+stressesIncrement;
            fb=self.damageFunction(newStressesB);
            if fb>=0
                isNonlinear=true;
                ab=NonLinearMaterial3D.verticalVectorToYieldSurface(↩
                    newStressesB);
                dl=fb/(ab'*C*ab);
                newStressesC=newStressesB-dl*C*ab;
                fc=self.damageFunction(newStressesC);

                ac=NonLinearMaterial3D.verticalVectorToYieldSurface(↩
                    newStressesC);
                r=newStressesC-(newStressesB-dl*C*ac);
                Q=(eye(6)+dl*C*NonLinearMaterial3D.aDerivativeToStresses(↩
                    newStressesC));
                lamda=(fc-ac'*(Q\r))/(ac'*(Q\C*ac));
                stressesIncrement=-(Q\r)-lamda*(Q\self.C6*ac);
                newStressesC=newStressesC+stressesIncrement;
                fc=self.damageFunction(newStressesC);
                newStresses=newStressesC;
            else
                isNonlinear=false;
                newStresses=newStressesB;
            end
        end
    end
```

2.5 The consistent tangent modular matrix

Simo and Taylor [15], and Runesson and Samuelson [14] derived a tangent modular matrix that is fully consistent with the backward-Euler integration algorithm. As a consequence of the 'consistency', the use of the consistent tangent modular matrix significantly improves the convergence characteristics of the overall equilibrium iterations if a Newton-Raphson scheme is used for the latter. Standard techniques would use the modular matrix of (2.9) which is 'inconsistent' with the backward-Euler integrations scheme and destroys the 'quadratic convergence' inherent in the Newton-Raphson method.

A consistent tangent modular matrix with the backward-Euler return discussed in (2.4.3) can be derived using the following equations. Differentiation of the standard backward-Euler expression (2.34) gives

$$\dot{\boldsymbol{\sigma}} = \mathbf{C}\dot{\boldsymbol{\epsilon}} - \dot{\lambda}\mathbf{C}\mathbf{a} - \Delta\lambda\mathbf{C}\frac{\partial \mathbf{a}}{\partial \boldsymbol{\sigma}}\dot{\boldsymbol{\sigma}} \tag{2.40}$$

where the last term is omitted from the derivation of the standard tangent modular matrix. From (2.40),

$$\dot{\boldsymbol{\sigma}} = \left(\mathbf{I} + \Delta\lambda\mathbf{C}\frac{\partial \mathbf{a}}{\partial \boldsymbol{\sigma}}\right)^{-1}\mathbf{C}\left(\dot{\boldsymbol{\epsilon}} - \dot{\lambda}\mathbf{a}\right) = \mathbf{Q}^{-1}\mathbf{C}\left(\dot{\boldsymbol{\epsilon}} - \dot{\lambda}\mathbf{a}\right) = \mathbf{R}\left(\dot{\boldsymbol{\epsilon}} - \dot{\lambda}\mathbf{a}\right) \tag{2.41}$$

where the $\mathbf{Q}$ matrix has appeared before in (2.37).

To remain on the yield surface, $\dot{f}$ should be zero, and hence from (2.7)

$$\mathbf{a}^{\intercal}\dot{\boldsymbol{\sigma}} = \mathbf{a}^{\intercal}\mathbf{R}\dot{\boldsymbol{\epsilon}} - \dot{\lambda}\mathbf{a}^{\intercal}\mathbf{R}\mathbf{a} = 0 \tag{2.42}$$

and

$$\dot{\boldsymbol{\sigma}} = \mathbf{C}_{\mathbf{ct}}\dot{\boldsymbol{\epsilon}} = \left(\mathbf{R} - \frac{\mathbf{R}\mathbf{a}\mathbf{a}^{\intercal}\mathbf{R}^{\intercal}}{\mathbf{a}^{\intercal}\mathbf{R}\mathbf{a}}\right)\dot{\boldsymbol{\epsilon}} \tag{2.43}$$

where $\mathbf{C}_{\mathbf{ct}}$ is the consistent tangent modular matrix.

Matlab Code 2.6: Function get consistent modular matrix

```
function Ct=getConsistentTangentModularMatrix(self,stresses)
    f=self.damageFunction(stresses);
    if f>=0
        a=NonLinearMaterial3D.verticalVectorToYieldSurface(↩
            stresses);
        dl=f/(a'*self.C6*a);
        Q=(eye(6)+dl*self.C6*NonLinearMaterial3D.↩
            aDerivativeToStresses(stresses));
        R=Q\self.C6;
        Ct=R*(eye(6)-a*a'*R/(a'*R*a));
    else
        Ct=self.C6;
    end
end
```

There are special formulations for two-dimensional situations [5, Chapter 6.8].

3 Non Linear Finite Element Analysis

3.1 Introduction

With the creation of a non-linear material, as discussed in Chapter (2), the element which encapsulates the material becomes a non-linear element. Given some strains or displacements (strains can be computed from displacements), the non-linear finite element can return its stresses or the reactions (again the reactions can be computed from the stresses) according to the material law that is chosen. In that way we can see that the pure displacement version of the finite element method is the most convenient spatial discretisation method for the majority of the applications of non-linear constitutive relations. Its formulation is simple, and allows for a straightforward implementation of complicated constitutive relations. Apart from the displacement control methods, there are also load control methods, and even more advanced ones, such as path-following methods, Quasi-Newton methods, and branch switching techniques at bifurcation points [13, Chapter 4].

3.2 Load control method

Generally, in static finite element analysis we require

$$\mathbf{f}_{ext} - \mathbf{f}_{int} = 0 \tag{3.1}$$

where $\mathbf{f}_{ext}$ are the external forces and $\mathbf{f}_{int}$ are the internal forces.

For static non linear analysis time plays no role. Yet, also then we need a parameter to order the sequence of events. For this reason we shall continue to use the concept of 'time' also in static mechanical processes to order the loading sequence. In particular, the concept of time can be employed to apply the external load in a number of loading steps (or increments). It would be possible to impose the entire external load $\mathbf{f}_{ext}$ in a single step, but this wouldn't be a sensible approach because of the following reasons:

1. The set of algebraic equations that arises from the discretisation of a non-linear continuum model is non-linear, thus necessitating the use of an iterative procedure for its solution. For very large loading steps, the case of imposing the entire load in one step being the extreme, it is usually difficult to obtain a properly converged solution, if a solution can be obtained at all. Indeed, the convergence radius is limited for most commonly used iterative procedures, including the Newton-Raphson method.

2. Experiments show that most materials exhibit path-dependent behaviour. This means that different values for the stress are obtained depending on the strain path that is followed. For instance, the resulting stress can be different when we first apply tension on a panel followed by a shear strain increment or when the same strain increments are imposed in the reverse order. Evidently, the structural behaviour can only be predicted correctly if the strain increments are relatively small, so that the strain path is followed as closely as possible.

Along this line of reasoning we decompose the vector of unknown stress components at time $t+\Delta t$, denoted by $\boldsymbol{\sigma}^{t+\Delta t}$ into a stress vector $\boldsymbol{\sigma}^{t}$ at time t when the stress components are known, and $\Delta\boldsymbol{\sigma}$ which contains the hitherto unknown components of the stress increment

$$\boldsymbol{\sigma}^{t+\Delta t} = \boldsymbol{\sigma}^{t} + \Delta\boldsymbol{\sigma} \tag{3.2}$$

Substituting this additive decomposition into Equation (3.1) we get

$$\mathbf{f}_{ext}^{t+\Delta t} - \mathbf{f}_{int}^{t+\Delta t} = 0 \tag{3.3}$$

Contrary to a linear system, the internal forces $\mathbf{f}_{int}^{t+\Delta t}$, derived from the displacements $\Delta\mathbf{a}$ by solving

$$\mathbf{K}\Delta\mathbf{a} = \mathbf{f}_{ext}^{t+\Delta t} - \mathbf{f}_{int}^{t} \tag{3.4}$$

where $\mathbf{K}$ which is the tangential stiffness matrix of the structure upon a small increment of loading, will not equal the external forces $\mathbf{f}_{ext}^{t+\Delta t}$ and hence the equilibrium of (3.3) will not be satisfied. This would lead to a 'drifting away' from the true equilibrium solution, especially if relatively large loadings steps are employed. This is why equilibrium iterations within each loading step should be added. In this incremental-iterative solution method a first estimate for the displacement increment $\Delta\mathbf{a}$ is made through

$$\Delta\mathbf{a}_1 = \mathbf{K}_0^{-1} r_0 \tag{3.5}$$

where

$$r_0 = \mathbf{f}_{ext}^{t+\Delta t} - \mathbf{f}_{int,0} \tag{3.6}$$

is the residual vector at the beginning of the load increment. The subscript 1 of $\Delta\mathbf{a}$ signifies that we deal with the estimate in the first iteration for the incremental displacement vector. Likewise, the subscript 0 of the internal force vector relates to the fact that this vector is calculated using the stresses at the beginning of the loading step, i.e. that are left behind at the end of the previous iteration ($\boldsymbol{\sigma}_0 = \boldsymbol{\sigma}^t$)

$$\mathbf{f}_{int,0} = \mathbf{f}_{int,0}(\boldsymbol{\sigma}_0)!!!! \tag{3.7}$$

Then from the displacement vector $\Delta\mathbf{a}_1$ the strain increment $\Delta\boldsymbol{\epsilon}_1$ is estimated, whereupon, using the stress-strain law, the stress increment $\Delta\boldsymbol{\sigma}_1$ is computed. The new stresses are now

$$\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_0 + \Delta\boldsymbol{\sigma}_1 \tag{3.8}$$

Generally, the new internal force vector $\mathbf{f}_{int,1}$ (that is computed from $\boldsymbol{\sigma}_1$) is not in equilibrium with the external loads $\mathbf{f}_{ext}^{t+\Delta t}$,so the new residual vector is computed

$$r_1 = \mathbf{f}_{ext}^{t+\Delta t} - \mathbf{f}_{int,1} \tag{3.9}$$

and then the correction to the displacement increment by $d\mathbf{a}_2$

$$d\mathbf{a}_2 = \mathbf{K}_1^{-1} r_1 \tag{3.10}$$

where $\mathbf{K}_1$ is the updated tangential stiffness matrix. The displacement increment after the second iteration in the loading step follows from

$$\Delta\mathbf{a}_2 = \Delta\mathbf{a}_1 + d\mathbf{a}_2 \tag{3.11}$$

In a similar fashion the next iterations are now proceeded. The process results in stresses that are in equilibrium internally and with the applied external loading within some user-prescribed convergence tolerance. A graphical explanation is given in Figure (3.1). The general algorithm for the load control method for each load step is:

1. Initialize the data for the loading step. Set $\Delta\mathbf{a}_0 = 0$

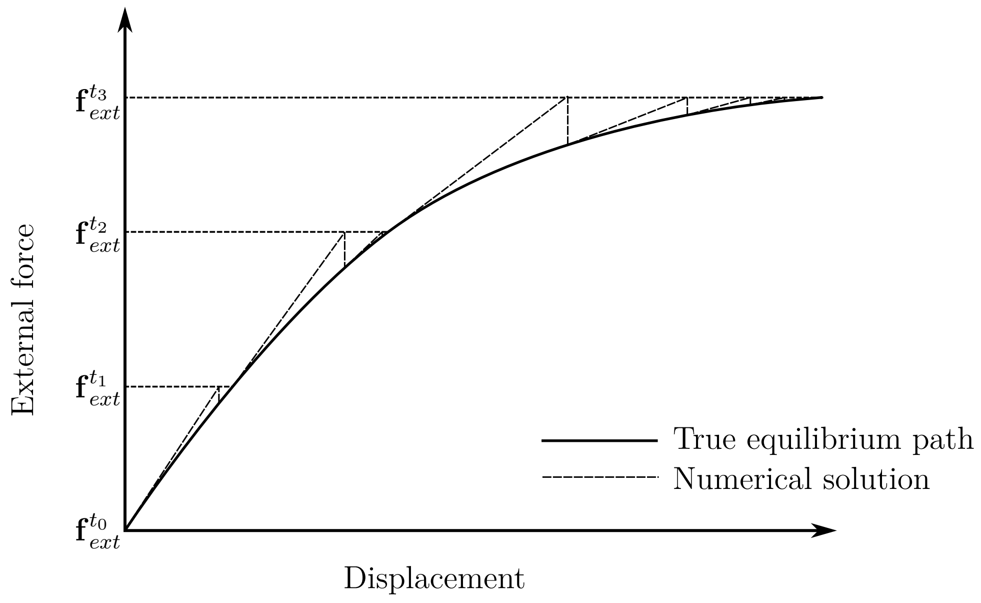


Figure 3.1: Graphical explanation of the 'load control' method

2. Compute the new external force vector $\mathbf{f}_{ext}^{t+\Delta t}$
3. Compute the tangential stiffness matrix $\mathbf{K_j}$ (see chapter 2)
4. Adjust for prescribed displacements and linear dependence relations
5. Solve the linear system $\mathbf{K_j} d\mathbf{a}_{j+1} = \mathbf{f}_{ext}^{t+\Delta t} - \mathbf{f}_{int,j}$
6. Add the correction $d\mathbf{a}_{j+1}$ to the incremental displacement vector (equation (3.11))
7. Compute the strain increment $\Delta\boldsymbol{\epsilon}_{i,j+1}$ for each integration point i
8. Compute the stress increment from the strain increment for each integration point i using the methods of chapter 2
9. Add the stress increment to $\boldsymbol{\sigma}_{i,0}$ for each integration point i
10. Compute the internal force vector from the stresses at gauss points
11. Check convergence with norm $\|\mathbf{f}_{ext}^{t+\Delta t} - \mathbf{f}_{int,j+1}\|$ and proceed to the next loading step or go to step 3

An issue that has not been discussed yet, is the implicit assumption that the tangential stiffness matrix $\mathbf{K_j}$ is updated after each iteration. Indeed, it can be rather costly to compute and to decompose a stiffness matrix every iteration, as is being done within a full Newton-Raphson process, especially in computations of three-dimensional structures. This has motivated the search for methods which obviate the need to construct and decompose a tangential stiffness matrix in every iteration. There are other load control methods that compute the stiffness matrix once in each load step and use the same or use linear modifications of that such as the Quasi-Newton techniques. [13, Chapter 4]

Matlab Code 3.1: Function 'NonLinearLoadControlAnalysis'

```
function nonLinearLoadControlAnalysis(ElementList,loadVector,steps)
            solverTime=tic;
            [pDOF prescribedDisplacements]=ElementList.nodeList.↩
                getPrescribedDisplacements;
            loadVectorStep=loadVector*(1/steps);
            fprintf('Non Linear Analysis for %5.0f steps \n',steps)
            for i=1:steps
                fprintf('Step %5.0f of %5.0f ... ',i,steps)
                newLoadVector=loadVectorStep*i;
                [newDisplacements newForces]=Solver.nonLinearNewtonRaphson↩
                    (ElementList,prescribedDisplacements,pDOF,↩
                    newLoadVector);
                ElementList.forces=newForces;
            end

            time=toc(solverTime);
            fprintf('Time for calculations %10.3f seconds\n',time)
        end
```

Matlab Code 3.2: Function 'nonLinearNewtonRaphson'

```
function [displacementVector forcesVector]=nonLinearNewtonRaphson(↩
    ElementList,prescribedDisplacements,prescribedDisplacementsDOF,↩
    loadVector)
                freeDOF=ElementList.getDOF;
                freeDOF(prescribedDisplacementsDOF)=[];
                fext=loadVector;
                fint=ElementList.getLoadsFromStresses;
                n=0;
                displacements=ElementList.displacements;
                if isempty(displacements)
                    dof=ElementList.getDOF;
                    displacements=zeros(dof,1);
                end
                remainingLoads=fext-fint;
                remainingLoads(prescribedDisplacementsDOF)=0;

                while norm(remainingLoads(freeDOF))/norm(loadVector)>↩
                    Solver.forcesAccuracy
                    n=n+1;
                    fprintf('Iteration %5.0f ... ',n)
                    ElementList.createStiffnessMatrix;
                    StiffnessMatrix=ElementList.getStiffnessMatrix;
                    [displacementStep forcesStep]=Solver.classicSolver(↩
                        StiffnessMatrix,prescribedDisplacements',↩
                        prescribedDisplacementsDOF,remainingLoads);
                    displacements=displacements+displacementStep;
                    %forces=forces+forcesStep;
                    ElementList.setDisplacements(displacements);
                    %ElementList.forces=forces;
                    ElementList.setStressesToGaussPoints;
                    fint=ElementList.getLoadsFromStresses;

                    remainingLoads=fext-fint;
                    remainingLoads(prescribedDisplacementsDOF)=0;
                    reverseStr = repmat(sprintf('\b'), 1, 20);
                    fprintf(reverseStr);
                end
                fprintf('Completed in %2.0f iterations\n',n)
                displacementVector=displacements;
                forcesVector=fint;
        end
```

3.3 Displacement control method

Alternatively to the previous section where we prescribe load increments, we can prescribe displacement increments. This so-called displacement control procedure causes a stress development within the specimen, which in turn results in nodal forces at the nodes where the displacements are prescribed. Summation of these forces gives the total reaction force, which, except for a minus sign, equals the equivalent external load that would be caused by the prescribed displacements. However, when there is no preference for either load or displacement control from a physical point of view, the latter method is often to be preferred. The reasons for the preference for displacement control are twofold

1. The tangential stiffness matrix is better conditioned for displacement control than for load control. This tends to result in a faster convergence behaviour of the iterative procedure.

2. Under load control, the tangential stiffness matrix becomes singular at a limit point in the load-deflection diagram, not only when global failure occurs, but also when we have a local maximum along this curve (Figure 3.2). The tangential stiffness matrix of the displacement controlled problem, on the other hand, does not become singular.

These statements are best elucidated starting from Equation (3.4). This equation has been derived for load control, and the prescribed external load

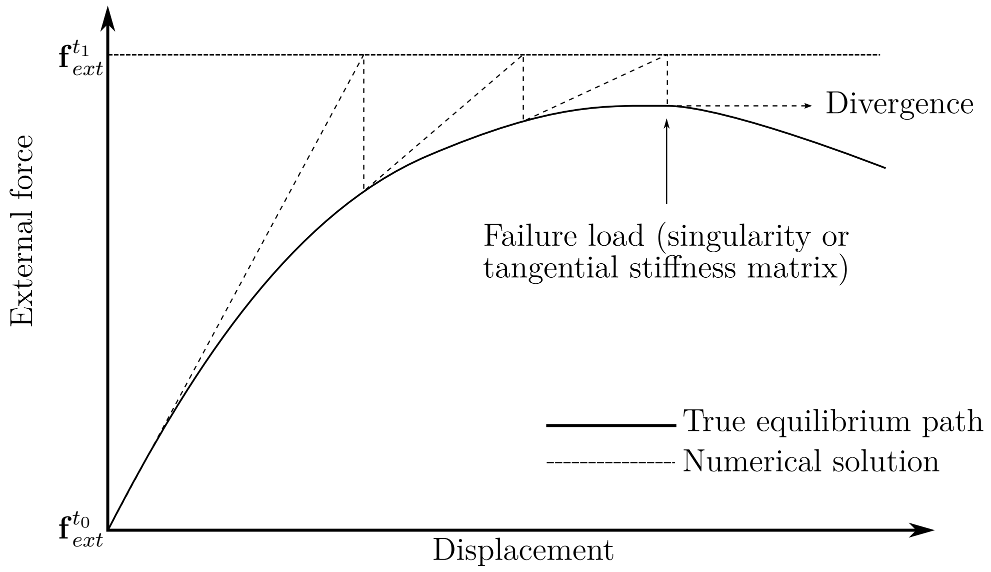


Figure 3.2: Singularity of tangential stiffness matrix at limit point and divergence of iterative procedure

level is contained explicitly in the vector $\mathbf{f}_{ext}$. The use of displacement control does not directly cause external forces to be exerted on the structure. Rather, a number of non-zero displacements are prescribed in an incremental loading programme. We now decompose the incremental displacement vector $\Delta\mathbf{a}$ into a vector that contains only degrees of freedom that are 'free', i.e. which have to be calculated, $\Delta\mathbf{a}_f$, and displacement increments that have been assigned a certain non-zero value, $\Delta\mathbf{a}_p$

$$\Delta\mathbf{a} = \begin{bmatrix} \Delta\mathbf{a}_f \\ \Delta\mathbf{a}_p \end{bmatrix} \tag{3.12}$$

In a similar manner the stiffness matrix can be partitioned, as

$$\mathbf{K} = \begin{bmatrix} \mathbf{K_{ff}} & \mathbf{K_{fp}} \\ \mathbf{K_{pf}} & \mathbf{K_{pp}} \end{bmatrix} \tag{3.13}$$

Using Equations (3.12) and (3.13), Equation (3.4) can be replaced by the expression

$$\begin{bmatrix} \mathbf{K_{ff}} & \mathbf{K_{fp}} \\ \mathbf{K_{pf}} & \mathbf{K_{pp}} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{a}_f \\ \Delta\mathbf{a}_p \end{bmatrix} = - \begin{bmatrix} \mathbf{f}_{f,int,0} \\ \mathbf{f}_{p,int,0} \end{bmatrix} \tag{3.14}$$

where it has been assumed that, apart from the prescribed displacements, no other forces act on the structure. Next, the unknown or 'free' displacement increments can be calculated by eliminating $\Delta\mathbf{a}_p$ from Equation (3.14). For the first iteration this elimination process yields

$$\Delta\mathbf{a}_{f,1} = -\mathbf{K_{ff}}^{-1}(\mathbf{K_{pf}}\Delta\mathbf{a}_p + \mathbf{f}_{f,int,0}) \tag{3.15}$$

while in the subsequent iterations the formula for computing the unknown degrees of freedom changes into

$$d\mathbf{a}_{f,j+1} = -\mathbf{K_{ff}}^{-1}\mathbf{f}_{f,int,0} \tag{3.16}$$

since $d\mathbf{a}_p$ vanishes. Comparison of Equations (3.4) and (2.56) shows, that for the first iteration the external load $\mathbf{f}_{ext}^{t+\Delta t}$ must be replaced by the 'equivalent force vector' $\mathbf{K_{pf}}\boldsymbol{\Delta}\mathbf{a_p}$ when switching from load to displacement control. In the next iterations this contribution vanishes altogether for displacement control.

Matlab Code 3.3: Function 'nonLinearDisplacementControlAnalysis'

```
function [displacementVector n]=nonLinearDisplacementControlAnalysis(↩
    ElementList,prescribedDisplacements,prescribedDisplacementsDOF)

            fprintf('Non Linear Displacement Control Analysis \n')

            % Displacements
            ElementList.createStiffnessMatrix;
            K=ElementList.getStiffnessMatrix;
            cDOF=prescribedDisplacementsDOF;
            prescribedDisp=prescribedDisplacements;
            Kff=K.deleteDOF(cDOF);
            fDOF=Kff.getDOF;
            Kfp=K.getKce(fDOF,cDOF);

            df=Kff.getStiffnessMatrix\(-Kfp.getStiffnessMatrix*↩
                prescribedDisp);

            %Reaction forces
            Kpp=K.getKcc(cDOF);
            Kpf=K.getKce(cDOF,fDOF);
            Rp=Kpf.getStiffnessMatrix*df+Kpp.getStiffnessMatrix*↩
                prescribedDisp;

            % Displacement vector
            disp=zeros(size(K.getDOF,2),1);
            disp(fDOF)=df;
            disp(cDOF)=prescribedDisp;
            displacementVector=disp;

            % Forces vector
            R=zeros(size(K.getDOF,2),1);
            R(cDOF)=Rp-R(cDOF);
            forcesVector=R;
```

```
        n=0;
        displacementStep=displacementVector;
        ElementList.setDisplacements(displacementVector);
        while norm(displacementStep)/norm(prescribedDisplacements)>↩
            Solver.displacementsAccuracy
            n=n+1;
            fprintf('Iteration %5.0f ... ',n)
            ElementList.setStressesToGaussPoints;
            fint=ElementList.getLoadsFromStresses;
            %ElementList.createStiffnessMatrix;
            StiffnessMatrix=ElementList.getStiffnessMatrix;
            Kff=StiffnessMatrix.deleteDOF(prescribedDisplacementsDOF);
            displacementStep=-(Kff.getStiffnessMatrix)\fint(fDOF);
            displacementVector(fDOF)=displacementVector(fDOF)+↩
                displacementStep;
            %displacementVector(fDOF)=displacementStep;
            ElementList.setDisplacements(displacementVector);
            reverseStr = repmat(sprintf('\b'), 1, 20);
            fprintf(reverseStr);
        end
        fprintf('Completed in %2.0f iterations\n',n)

    end
```

4 The non-intrusive global local strategy

4.1 Introduction

In this chapter, the main analysis technique for the handling of the local non-linearities will be discussed. This technique is based upon a simple idea. Instead of changing the linear elastic model that we have, by changing the mesh and the type of finite elements in the region that the non-linearity is expected, a second model is created only for that region with the geometric details, mesh refinements or specific constitutive laws that are required. The first model, the linear elastic one,which is referred as global, stays unchanged (it contains also the region of the second model with linear elastic properties). Then with a coupling iterative algorithm between the two models, by transferring displacements and forces from one model to the other and after various analyses, the exact solution is achieved [8]. The global model reaches the solution for the linear elastic part while the local model for the non-linear region.

This method is designed to be both exact and non-intrusive. It is an exact solution as it reaches the solution of a model with elastic linear properties in general and the non-linear properties at the chosen local region. In other words, if the local region is selected correctly, as the only region where non-linearities occur, then the solution of this strategy reaches the solution of a full non-linear model. The great advantage of this local handling is highlighted in its low computational cost. The method is also non-intrusive which means that the solvers as well as the models are not meant to be manipulated in a way that would require intervention in standard fem programs (even commercial). This means that the two models, the global and the local, can be created in any fixed commercial program that has the appropriate libraries (of finite elements, materials etc.) for the local non-linearity.

In addition, all the information that is required for the coupling between the two models is data that is always found from the standard fem programs, such as displacements and forces. Moreover, the solvers that the strategy uses are solvers that FEM programs already have and there is not any need to be changed. The only algorithm that is needed to be implemented is the one that makes the information exchange between the two models.

Another significant specification here is that, in addition to handling local non-linearity, the local model can be used to introduce geometric details, mesh refinements or specific constitutive laws that are absent from the global model. Furthermore, it can also be analyzed using a separate piece of code, which may contain features that are not included in the global FE solver. In this sense, the approach also works as a flexible and exact structural reanalysis and solver coupling technique.

4.2 The reference problem and the two models

We consider a static mechanical problem set on a domain Ω. We also assume that the behaviour all of the domain is linear and elastic except in a small region, that we call Ω_I. In this region, Ω_I, we assume that the constitutive law is elastic-plastic, although, as already discussed, other forms of non-linearity and geometry abnormalities could as well be considered. We also assume that the region, Ω_I, in which the elastic-plastic behaviour occurs is already known and that the rest of the region,which is called complement area, $\Omega_C = \Omega \backslash \Omega_I$, will remain purely elastic and linear. Only one loading increment is considered, but multiple load increments are handled by repeating the procedure in an incremental scheme. [1] The reference problem is shown in Figure 8.2 with its boundary conditions and loads.

This problem is rephrased in a splitted way by creating two models, the global model and the local model. The global model is a simplified linear elastic model (Figure 4.2a). This model represents the entire structure and contains the elastic-plastic region but with linear elastic properties. The local model describes only the non-linear region (Figure 4.2b). This reminds of traditional submodeling techniques with one major difference: the local model can contain many enhancements or changes without causing inaccuracies in the final result. To define the local model, a proper model error indicator is being used after a first global linear analysis. At this point, it should be clarified that in the sake of simplicity the following assumptions are made:

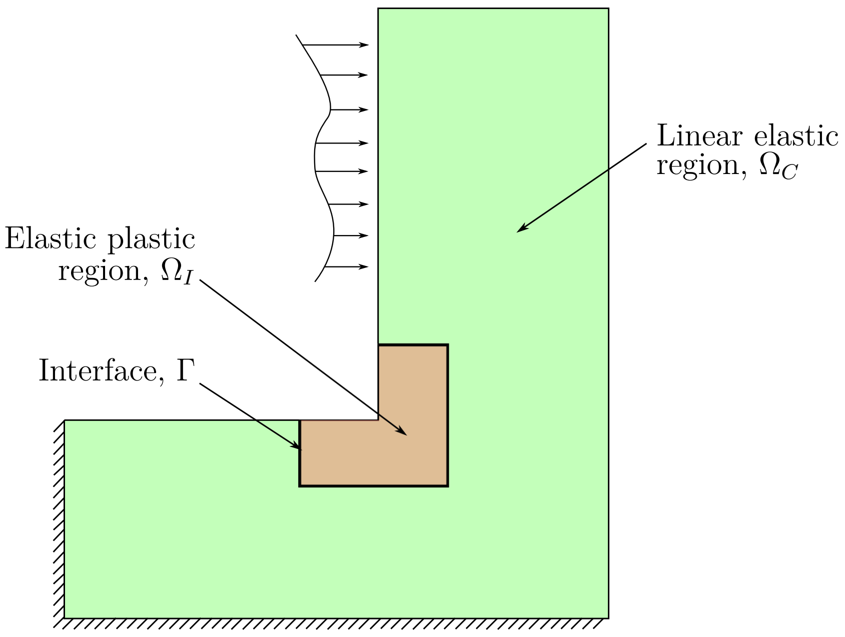
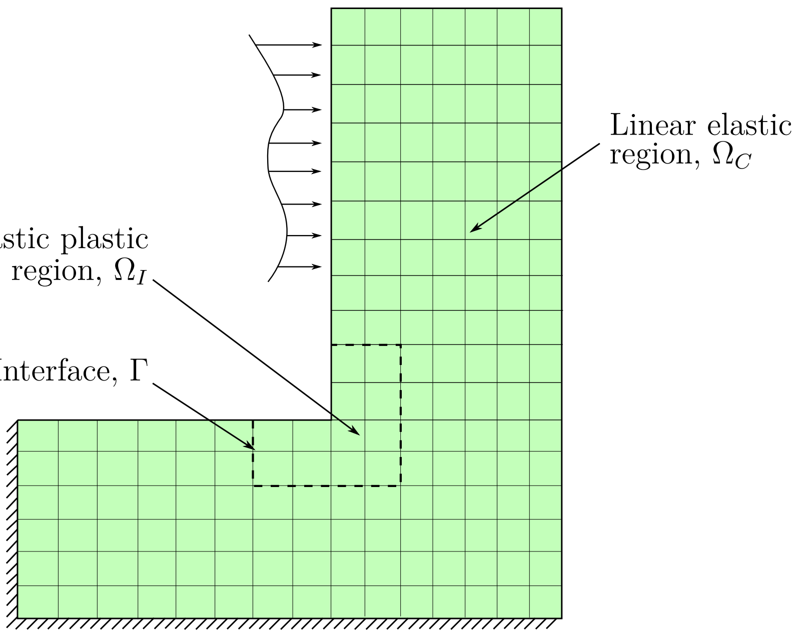

Linear elastic region, Ω_C
Elastic plastic region, Ω_I
Interface, Γ

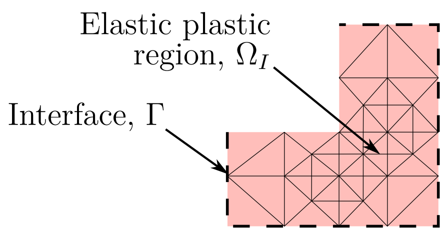
Figure 4.1: The reference problem

1. Ω_C is a set of uncut elements of the global mesh (Γ doesn't cut any global element)
2. Ω_I is the exact region covered by the local mesh, and Γ is part of its boundary
3. both meshes match on Γ : same nodes, edges, faces, degrees of freedom and shape or basis functions. Therefore there is no need for a separate discretization of Γ. Non-conforming interfaces have not been considered so far, though a standard mortar technique [3] should provide an effective answer to this problem.

In addition, the local model completely replaces the global model in the non-linear region. In other terms, the two models affect each other only on the boundary of the non-linear region. There is no need for model compatibility inside Ω_I. The two representations of Ω_I can be completely different in terms of geometries, discretizations and constitutive laws, given that the meshes match on Γ. This property makes it easier for the user to define the local mesh and allows the method to be used for structural reanalysis and solver coupling purposes, as explained previously.

The logic of the strategy is that the final solution takes into account the local model for the non-linear region and the global model for the rest area


Linear elastic region, Ω_C
Elastic plastic region, Ω_I
Interface, Γ
(a) Global model



(b) Local model

Figure 4.2: The two models of the strategy

(Figure 4.3). Hence, the displacement field, $\mathbf{u}$, of the solution is defined as

$$\mathbf{u} = \begin{cases} \mathbf{u}^L(\mathbf{x}) & \text{if } \mathbf{x} \in \Omega_I \\ \mathbf{u}^G(\mathbf{x}) & \text{if } \mathbf{x} \in \Omega_C \end{cases}$$

while the stress field, $\boldsymbol{\sigma}$, as

$$\boldsymbol{\sigma} = \begin{cases} \boldsymbol{\sigma}^L(\mathbf{x}) & \text{if } \mathbf{x} \in \Omega_I \\ \boldsymbol{\sigma}^G(\mathbf{x}) & \text{if } \mathbf{x} \in \Omega_C \end{cases}$$

The 'global-local' solution verifies every equation of the reference problem if, and only if, the following three conditions are satisfied [7]:

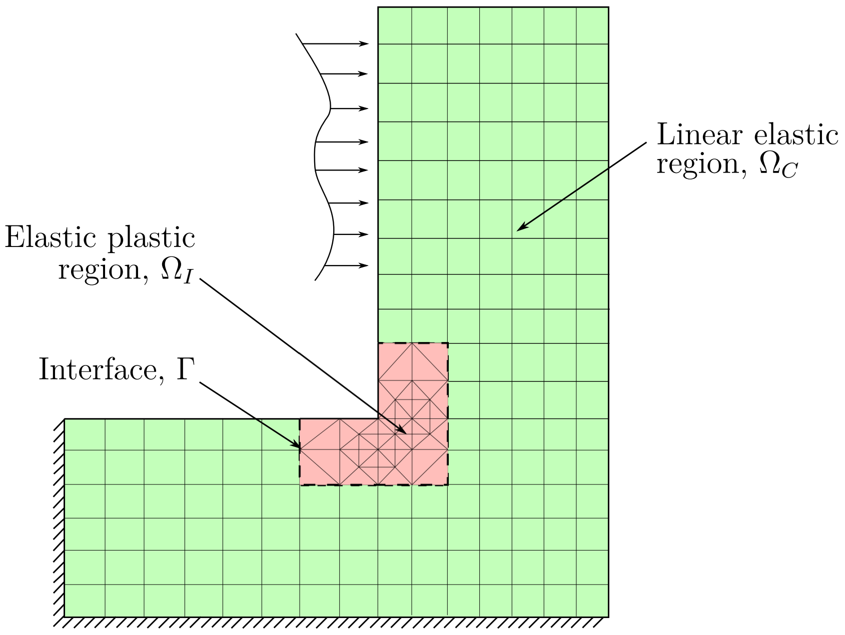

Linear elastic region, Ω_C
Elastic plastic region, Ω_I
Interface, Γ

Figure 4.3: The reference model

1. the local solution, $\mathbf{u}^L$ and $\boldsymbol{\sigma}^L$, verifies each equation written in Ω_I and on $\partial\Omega_I \backslash \Gamma$ (equilibrium, boundary conditions and elastic-plastic constitutive equations)
2. the restriction of the global solution to Ω_C, $\mathbf{u}_C^G$ and $\boldsymbol{\sigma}_C^G$, verifies each equation written in Ω_C and on $\partial\Omega_C \backslash \Gamma$ (equilibrium, boundary conditions and linear elastic constitutive equations)
3. both solutions match on Γ (displacements are equal and tractions are balanced)

These three conditions form the so-called global/local formulation. They are similar to the formulation of non-overlapping domain decomposition methods: the overlap of models is eliminated from the equations.

4.3 Equations of the problem

According to our previous hypothesis, the system that corresponds to the global problem has the classic form

$$\mathbf{K}^{\mathbf{G}}\mathbf{u}^G = \mathbf{f}^G \tag{4.1}$$

where $\mathbf{K^G}$, $\mathbf{u}^G$ and $\mathbf{f}^G$ are the stiffness matrix, the displacement field vector and the forces vector of the global model. Decomposing equation (4.1) leads to

$$\begin{bmatrix} \mathbf{K^G_{CC}} & \mathbf{K^G_{C\Gamma}} & 0 \\ \mathbf{K^G_{\Gamma C}} & \mathbf{K^G_{\Gamma\Gamma,C}} + \mathbf{K^G_{\Gamma\Gamma,I}} & \mathbf{K^G_{\Gamma I}} \\ 0 & \mathbf{K^G_{I\Gamma}} & \mathbf{K^G_{II}} \end{bmatrix} \begin{bmatrix} \mathbf{u}^G_C \\ \mathbf{u}^G_\Gamma \\ \mathbf{u}^G_I \end{bmatrix} = \begin{bmatrix} \mathbf{f}^G_C \\ \mathbf{f}^G_{\Gamma,C} + \mathbf{f}^G_{\Gamma,I} \\ \mathbf{f}^G_I \end{bmatrix} \tag{4.2}$$

where the subscripts I, C and Γ represent respectively the degrees of freedom of the areas of Ω_I, Ω_C and the interface Γ, while the contribution of the elements of Ω_I and Ω_C on the interface's Γ stiffness matrix has been separated with the subscripts I and C.

On the other hand, the non-linear system that corresponds to the local problem without the boundary conditions on Γ and therefore incomplete takes the form

$$\mathbf{g}^L(\mathbf{u}^L) = \mathbf{f}^L + \boldsymbol{\lambda}^L \tag{4.3}$$

where $\mathbf{g}$ denotes the vector of nodal internal forces, which is a non-linear function of $\mathbf{u}$, and $\boldsymbol{\lambda}$ denotes the vector of the nodal forces corresponding to the boundary condition on unspecified Γ. Decomposing equation (4.3) for the degrees of freedom of the interface, Γ, and the the interior, Ω_I, we get

$$\begin{bmatrix} \mathbf{g}^L_\Gamma(\mathbf{u}^L_\Gamma, \mathbf{u}^L_I) \\ \mathbf{g}^L_I(\mathbf{u}^L_\Gamma, \mathbf{u}^L_I) \end{bmatrix} = \begin{bmatrix} \mathbf{f}^L_\Gamma \\ \mathbf{f}^L_I \end{bmatrix} + \begin{bmatrix} \boldsymbol{\lambda}^L_\Gamma \\ 0 \end{bmatrix} \tag{4.4}$$

Note that, there is no link between the local and the global degrees of freedom inside Ω_I; as a result the two discretizations are independent. However, the degrees of freedom of the interface, Γ are the same for both models.

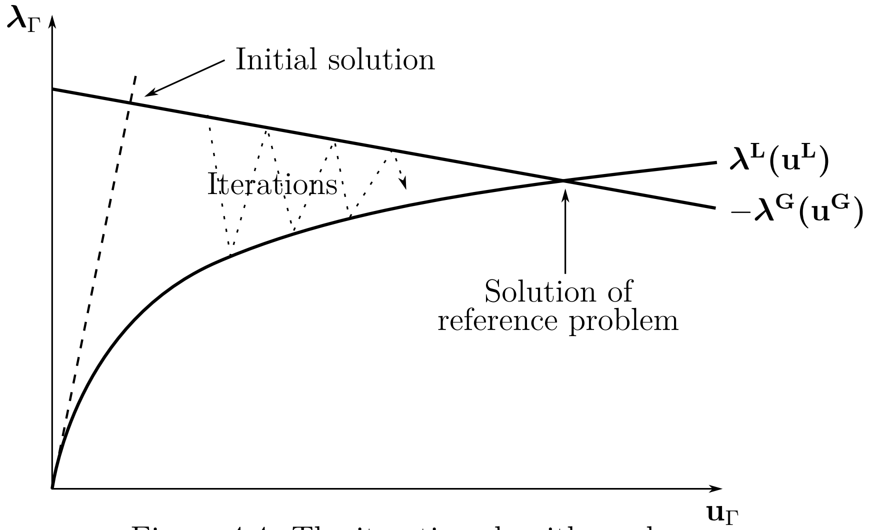
Finally, the finite element formulation of the reference problem is defined by substituting in the global decomposed formulation (equation (4.2)) all the contributions of Ω_I with the local decomposed formulation of equation (4.4). Hence the problem takes the form

$$\begin{bmatrix} \mathbf{K^G_{CC}} & \mathbf{K^G_{C\Gamma}} & 0 \\ \mathbf{K^G_{\Gamma C}} & \mathbf{K^G_{\Gamma\Gamma,C}} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}^R_C \\ \mathbf{u}^R_\Gamma \\ \mathbf{u}^R_I \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{g}^L_\Gamma(\mathbf{u}^R_I, \mathbf{u}^R_\Gamma) \\ \mathbf{g}^L_I(\mathbf{u}^R_I, \mathbf{u}^R_\Gamma) \end{bmatrix} = \begin{bmatrix} \mathbf{f}^G_C \\ \mathbf{f}^G_{\Gamma,C} + \mathbf{f}^L_{\Gamma,I} \\ \mathbf{f}^L_I \end{bmatrix} \tag{4.5}$$

where superscript R represents the solution of the reference problem (in the finite element formulation) as shown in figure 4.3.

4.4 The exchange algorithm

After a first linear analysis to the global model, and after defining the local area and creating the local model there are two variations of the exchange algorithm depending on the information being exchanged between the two models.



$\boldsymbol{\lambda}_\Gamma$
Initial solution
Iterations
$\boldsymbol{\lambda}^{\mathbf{L}}(\mathbf{u}^{\mathbf{L}})$
$-\boldsymbol{\lambda}^{\mathbf{G}}(\mathbf{u}^{\mathbf{G}})$
Solution of reference problem
$\mathbf{u}_\Gamma$

Figure 4.4: The iterative algorithm scheme

4.4.1 The displacements variation

In the displacement variation, in every iteration, we prescribe to the local model the continuity of displacements. The algorithm of this variation is the following

1. **Local Analysis.** The displacements of the interface Γ, that have been calculated from the global linear analysis, are prescribed at the local model (on Γ).

$$\mathbf{u}_\Gamma^L = \mathbf{u}_\Gamma^G$$

   Then local problem is solved with the prescribed displacements as in equation (4.4). For the elastic-plastic elements of the current thesis, this means that a 'displacement control' analysis is being driven as described in chapter 3.

2. **Calculation of the residual.** As the continuity of the displacements has been achieved with the previous step, the residual is calculated by taking into account the lack of equilibrium of forces on the interface Γ. Thus the residual is the sum of the interface's nodal forces of the surrounding elements. It takes the form

$$\mathbf{r}_\Gamma = -(\boldsymbol{\lambda}_\Gamma^L + \boldsymbol{\lambda}_{\Gamma,C}^G) \tag{4.6}$$

where $\boldsymbol{\lambda}_\Gamma^L$ is the nodal forces of the local model on the interface Γ after the local non-linear analysis (by solving equation (4.4)), and $\boldsymbol{\lambda}_{\Gamma,C}^G$ are the nodal forces of the global model, only from the complement area and not from the non-linear region (Ω_I).

3. **Global correction.** The global model is loaded with the calculated residual and all other prescribed displacements and loads are set to zero. From this analysis a corrective term $\Delta\mathbf{u}^G$ is computed that represents the effect of the non-linear region to the global model.

$$\mathbf{K}^G \Delta\mathbf{u}^G = \mathbf{r}^G = \begin{bmatrix} 0 \\ \mathbf{r}_\Gamma \\ 0 \end{bmatrix} \tag{4.7}$$

then global solution is updated by adding this correction term

$$\mathbf{u}^G \leftarrow \mathbf{u}^G + \Delta\mathbf{u}^G \tag{4.8}$$

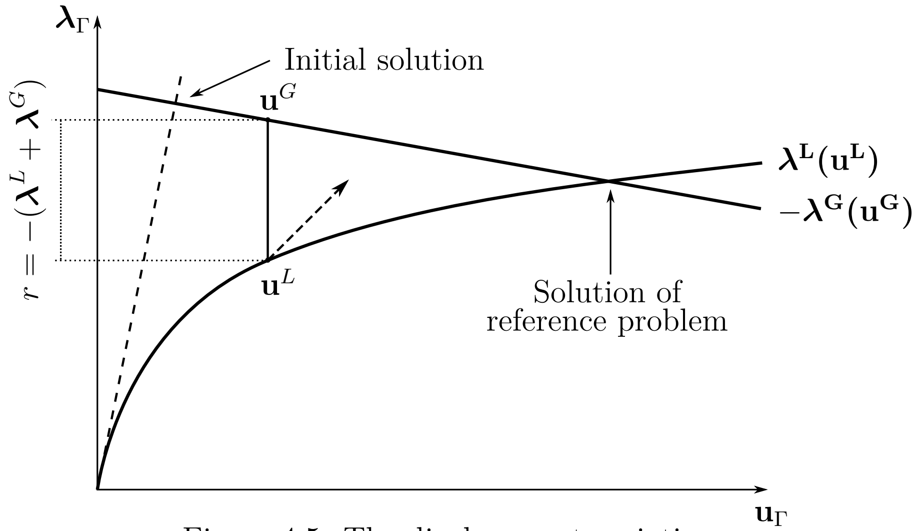

$\boldsymbol{\lambda}_\Gamma$
Initial solution
$\mathbf{u}^G$
$r = -(\boldsymbol{\lambda}^L + \boldsymbol{\lambda}^G)$
$\boldsymbol{\lambda}^\mathbf{L}(\mathbf{u}^\mathbf{L})$
$-\boldsymbol{\lambda}^\mathbf{G}(\mathbf{u}^\mathbf{G})$
$\mathbf{u}^L$
Solution of reference problem
$\mathbf{u}_\Gamma$

Figure 4.5: The displacement variation

4.4.2 The mixed boundary conditions variation

The term 'mixed boundary condition' can have several meanings. In this thesis, the use of Robin boundary conditions has been chosen. In other words, when we solve the local problem (4.4), we do not prescribe the continuity of displacements (the case of the previous variation) or the equilibrium of forces (that could be another variation), but a linear combination of these two equations. That is

$$(\boldsymbol{\lambda}_\Gamma^L + \boldsymbol{\lambda}_{\Gamma,C}^G) + \mathbf{A}(\mathbf{u}_\Gamma^L - \mathbf{u}_\Gamma^G) = 0 \tag{4.9}$$

where $\mathbf{A}$ is a parameter of the method, but also a square matrix representing an interface stiffness. Likewise the residual is also computed as a linear combination of the continuity of the displacements and the equilibrium of forces

$$(\boldsymbol{\lambda}_\Gamma^L + \boldsymbol{\lambda}_{\Gamma,C}^G) - \mathbf{B}(\mathbf{u}_\Gamma^L - \mathbf{u}_\Gamma^G) = 0 \tag{4.10}$$

where $\mathbf{B}$ is also a parameter of the method. It should be clear that Equations (4.9) and (4.10) are equivalent to the equilibrium of forces and displacements on the interface if, and only if, the matrix $[\mathbf{A} + \mathbf{B}]$ is invertible. If this condition is verified and if the algorithm is convergent, then the solution will converge to the reference solution. Apart from this condition, there is no restriction to the choice of $\mathbf{A}$ and $\mathbf{B}$, but this will be discussed later.

The algorithm of this variation is the following

1. **Local Analysis.** Using the mixed boundary conditions of equation (4.9) the local analysis of equation (4.4) takes the following formulation

$$\begin{bmatrix} \mathbf{g}_\Gamma^L(\mathbf{u}_\Gamma^L, \mathbf{u}_I^L) \\ \mathbf{g}_I^L(\mathbf{u}_\Gamma^L, \mathbf{u}_I^L) \end{bmatrix} + \begin{bmatrix} \mathbf{A} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_\Gamma^L \\ \mathbf{u}_I^L \end{bmatrix} = \begin{bmatrix} \mathbf{f}_\Gamma^L \\ \mathbf{f}_I^L \end{bmatrix} + \begin{bmatrix} -\boldsymbol{\lambda}_{\Gamma,C}^G + \mathbf{A}\mathbf{u}_\Gamma^G \\ 0 \end{bmatrix} \tag{4.11}$$

   It is clear that the matrix $\mathbf{A}$ represents a stiffness addition to the interface, while the mixed quantity $-\boldsymbol{\lambda}_{\Gamma,C}^G + \mathbf{A}\mathbf{u}_\Gamma^G$ is the additional load to the interface. Matrix $\mathbf{A}$ simulates the stiffness of the rest of the structure and thus makes the mixed formulation more realistic than the displacement variation. As we apply the mixed formulation in the form of a force vector, a 'load control' analysis method is needed for the local model as it is described in chapter 3.

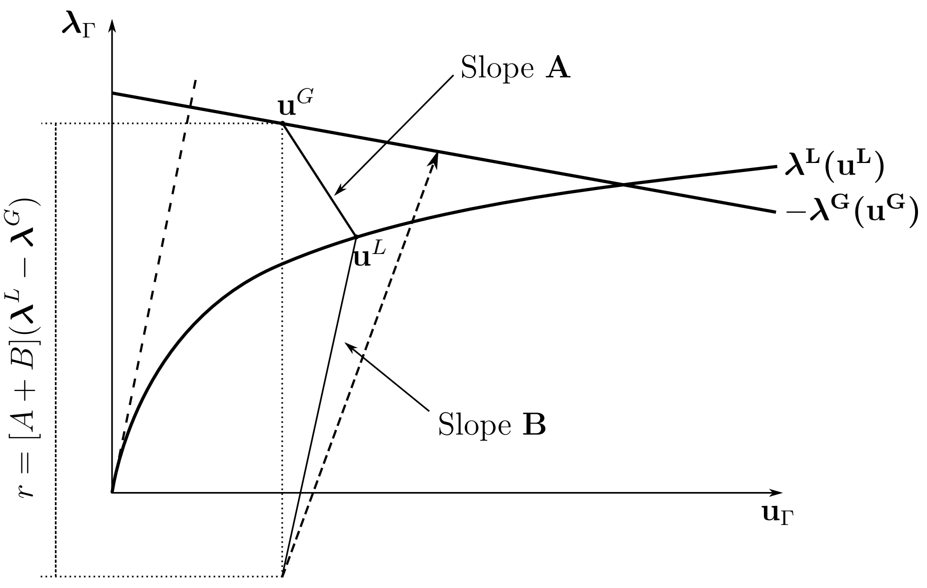
2. **Calculation of the residual.** The residual is calculated according to equation (4.10)

$$\mathbf{r}_\Gamma = -(\boldsymbol{\lambda}_\Gamma^L + \boldsymbol{\lambda}_{\Gamma,C}^G) + \mathbf{B}(\mathbf{u}_\Gamma^L - \mathbf{u}_\Gamma^G) \tag{4.12}$$

   while using using equation (4.9) it takes the form

$$\mathbf{r}_\Gamma = [\mathbf{A} + \mathbf{B}](\mathbf{u}_\Gamma^L - \mathbf{u}_\Gamma^G) \tag{4.13}$$

3. **Global correction.** There is no difference for the global correction in this variation. The global model is loaded with the calculated residual, while all other prescribed displacements and loads are set to zero. Then the corrective term $\Delta\mathbf{u}^G$ is computed and the global solution is updated (equation (4.8)).



$\boldsymbol{\lambda}_\Gamma$
Slope **A**
$\mathbf{u}^G$
$\boldsymbol{\lambda}^{\mathbf{L}}(\mathbf{u}^{\mathbf{L}})$
$-\boldsymbol{\lambda}^{\mathbf{G}}(\mathbf{u}^{\mathbf{G}})$
$\mathbf{u}^L$
$r = [A+B](\boldsymbol{\lambda}^L - \boldsymbol{\lambda}^G)$
Slope **B**
$\mathbf{u}_\Gamma$

Figure 4.6: The mixed variation

4.5 Interface formulation

It is convenient to write the equations in each model in condensed form, i.e. using only interface quantities. The global/local formulation can then be translated directly. The global solution in condensed form is

$$\left[\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{I}} + \mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}\right] \mathbf{u}^G_\Gamma - \left(\mathbf{b}^G_{\Gamma,I} + \mathbf{b}^G_{\Gamma,C}\right) = 0 \tag{4.14}$$

where $\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{I}}$ denotes the Schur complement of the complement area's stiffness matrix, and $\mathbf{b}^G_{\Gamma,C}$ the corresponding condensed right-hand side vector. More precisely

$$\begin{aligned}
\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{I}} &= \mathbf{K}^{\mathbf{G}}_{\mathbf{\Gamma\Gamma},\mathbf{I}} - \mathbf{K}^{\mathbf{G}}_{\mathbf{\Gamma I}}\mathbf{K}^{\mathbf{G}}_{\mathbf{II}}{}^{-1}\mathbf{K}^{\mathbf{G}}_{\mathbf{I\Gamma}} \\
\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}} &= \mathbf{K}^{\mathbf{G}}_{\mathbf{\Gamma\Gamma},\mathbf{C}} - \mathbf{K}^{\mathbf{G}}_{\mathbf{\Gamma C}}\mathbf{K}^{\mathbf{G}}_{\mathbf{CC}}{}^{-1}\mathbf{K}^{\mathbf{G}}_{\mathbf{C\Gamma}} \\
\mathbf{b}^{G}_{\Gamma,I} &= \mathbf{f}^{G}_{\Gamma,I} - \mathbf{K}^{\mathbf{G}}_{\mathbf{\Gamma I}}\mathbf{K}^{\mathbf{G}}_{\mathbf{II}}{}^{-1}\mathbf{f}^{G}_{I} \\
\mathbf{b}^{G}_{\Gamma,C} &= \mathbf{f}^{G}_{\Gamma,C} - \mathbf{K}^{\mathbf{G}}_{\mathbf{\Gamma C}}\mathbf{K}^{\mathbf{G}}_{\mathbf{CC}}{}^{-1}\mathbf{f}^{G}_{C}
\end{aligned}$$

or we can write $\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma}}\mathbf{u}^{G}_{\Gamma} - \mathbf{b}^{G}_{\Gamma} = 0$ where

$$\begin{aligned}
\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma}} &= \mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{I}} + \mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}} \\
\mathbf{b}^{G}_{\Gamma} &= \mathbf{b}^{G}_{\Gamma,I} + \mathbf{b}^{G}_{\Gamma,C}
\end{aligned}$$

The global solution, as it has to verify equations on Ω_C, takes also the form

$$\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{I}}\mathbf{u}^{G}_{\Gamma} - \mathbf{b}^{G}_{\Gamma,C} = \boldsymbol{\lambda}^{G}_{\Gamma,C} \tag{4.15}$$

The local problem with condensation at the interface takes the form

$$\mathbf{h}^{L}_{\Gamma}(\mathbf{u}^{L}_{\Gamma}) = \boldsymbol{\lambda}^{L}_{\Gamma} \tag{4.16}$$

where $\mathbf{h}^{L}_{\Gamma}$ is the local problem's discrete Steklov–Poincaré operator, i.e. the interface stiffness operator giving the reaction forces $\boldsymbol{\lambda}^{L}_{\Gamma}$ as a function of the interface displacements $\mathbf{u}^{L}_{\Gamma}$. A different notation is used because this group of equations is non-linear. Finally, interface displacements must match and interface nodal reaction forces must be balanced

$$\mathbf{u}^{G}_{\Gamma} = \mathbf{u}^{L}_{\Gamma} \tag{4.17}$$

$$\boldsymbol{\lambda}^{G}_{\Gamma,C} + \boldsymbol{\lambda}^{L}_{\Gamma} = 0 \tag{4.18}$$

This formulation is very similar to non-linear domain decomposition methods [12] [4]. The difference is that the global model is never substructured since the approach is non-intrusive; therefore, the complement area is never separated from the area of interest in the global model, and the quantities $\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}$ and $\mathbf{b}^{G}_{\Gamma,C}$ cannot be accessed directly. Equivalently, we can say that it is impossible to prescribe tractions directly on the complement area through Γ; only traction discontinuities between Ω_C and Ω_I can be prescribed in the global model.

The condensed form of the reference problem is

$$\mathbf{h}^{L}_{\Gamma}(\mathbf{u}^{R}_{\Gamma}) + \mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}\mathbf{u}^{R}_{\Gamma} - \mathbf{b}^{G}_{\Gamma,C} = 0 \tag{4.19}$$

5 The Schur approximation

5.1 Introduction

In section 4.4, two variations of the exchange algorithm were presented. The one was the displacements variation and the other the mixed variation. This two variations differ on the boundary conditions that are being exchanged. The displacements variation is easy to implement and creates a continuous displacement field. This is why it is very popular and often used in global/local or multi-level FEA, as well as hierarchical techniques.!! However, displacement boundary conditions suffer from a couple of limitations. First, as there is a difference between the stiffness of the local model and the stiffness of the global model at the area of interest, there is a great influence on the displacement field, which causes the boundary condition to be inaccurate. In other terms, during the first iterations, global interface displacements are usually quite different from reference interface displacements. This affects the solution's accuracy and, therefore, the algorithm's convergence. A common workaround is to position the interface far enough from the local details, but this increases the size of the local problem unnecessarily. Another problem is that if plasticity occurs at integration points located close to the boundary, we have noticed that the prescribed displacements can cause convergence difficulties. Mixed boundary conditions are commonly used in domain decomposition methods for these reasons. They are known to be quite insensitive to the difference of stiffnesses and give accurate results from the first iterations, which implies fast convergence, and they do not have a tendency to cause stress concentrations or convergence problems. Their increased accuracy was also shown in the field of global/local analysis. Therefore, they seemed to be a natural choice for the method.

5.2 The interface stiffness

As discussed in subsection 4.4.2 prescribing mixed boundary conditions to the interface takes the form

$$(\boldsymbol{\lambda}_\Gamma^L + \boldsymbol{\lambda}_{\Gamma,C}^G) + \mathbf{A}(\mathbf{u}_\Gamma^L - \mathbf{u}_\Gamma^G) = 0 \tag{5.1}$$

In addition, from the interface formulation of section 4.5 we have the following equations. The local problem is

$$\mathbf{h}_\Gamma^L(\mathbf{u}_\Gamma^L) = \boldsymbol{\lambda}_\Gamma^L \tag{5.2}$$

the global solution to the complement area is

$$\mathbf{S}_{\mathbf{\Gamma},\mathbf{C}}^{\mathbf{G}}\mathbf{u}_\Gamma^G = \mathbf{b}_{\Gamma,C}^G + \boldsymbol{\lambda}_{\Gamma,C}^G \tag{5.3}$$

the continuity of the displacements and the equilibrium of forces

$$\mathbf{u}_\Gamma^G = \mathbf{u}_\Gamma^L \tag{5.4}$$

$$\boldsymbol{\lambda}_{\Gamma,C}^G + \boldsymbol{\lambda}_\Gamma^L = 0 \tag{5.5}$$

and the reference problem

$$\mathbf{h}_\Gamma^L(\mathbf{u}_\Gamma^R) + \mathbf{S}_{\mathbf{\Gamma},\mathbf{C}}^{\mathbf{G}}\mathbf{u}_\Gamma^R = \mathbf{b}_{\Gamma,C}^G \tag{5.6}$$

By replacing in equation (5.1) the term $\boldsymbol{\lambda}_\Gamma^L$ with its equal form equation (5.2) we take the form

$$\mathbf{h}_\Gamma^L(\mathbf{u}_\Gamma^L) + \mathbf{A}\mathbf{u}^L = \mathbf{A}\mathbf{u}^G - \boldsymbol{\lambda}_C^G \tag{5.7}$$

In other terms, the interface stiffness $\mathbf{A}$ is assembled into the local problem and a mixed load vector $\mathbf{A}\mathbf{u}^G - \boldsymbol{\lambda}_C^G$ is prescribed. This shows that the Robin condition can be a powerful tool for modeling the stiffness of the rest of the structure, that is the complement domain, as 'seen from outside'. More specifically by comparing with equation (5.6), it is easy to see that if $\mathbf{A}$ is equal to the Schur complement of the complement domain $\mathbf{S}_{\mathbf{\Gamma},\mathbf{C}}^{\mathbf{G}}$, then the mixed local problem (5.7) becomes equivalent to the reference problem (5.6). Therefore, if we could compute $\mathbf{S}_{\mathbf{\Gamma},\mathbf{C}}^{\mathbf{G}}$ exactly, the analysis strategy would be:

1. exact, which means that the local solution will be the exact reference solution

2. direct, which means that with the first local analysis we would reach the reference solution and hence, no other iterations would be needed

3. insensitive to the interface position, unlike the variation with the prescribed displacements

Unfortunately, to compute $\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}$ means to perform static condensation on the complement area. Of course this technique is known to be extremely expensive especially on problems with many degrees of freedom, such as big 3D problems. Maybe there is a possibility here for this technique (of the Schur complement) to be combined with domain decomposition techniques, such as FETI, and reduce the computation cost at accepted levels. Furthermore, this static condensation is also intrusive. To condensate the complement area we would have to create a model of this area and this is an intrusion to the linear model.

This chapter proposes that an alternative is the approximation of the Schur complement. This approximation should of course be inexpensive to compute, and at the same time be 'close enough' to the exact value so that fast convergence is obtained. In addition, since the method is aimed to be non-intrusive, the building of the approximation should work on irregular meshes and require as little user intervention (such as manual mesh operations) as possible. There are many approximations that have been suggested, but the most common are the following two types of approximation. Approximations based on the elements on the vicinity of the interface, which we can call short-scale approximations, and approximations that are inspired from homogenization techniques, which we can call large-scale approximations (we use large-scale interface loads and displacements in these approximations). In this chapter this two approximations will be reviewed and then a combination of the above approximations will be discussed. This 'two-scale approximation' is proposed by Genre et. al [7] and is a technique that was used at this thesis.

5.3 Short-scale approximation

These short-scale techniques are also called element strip techniques. As the name suggests, these techniques consist in performing exact static condensation, not on the whole complement area, but on a strip of elements adjacent to the interface, usually clamped at the other end. This leads to much lower memory requirements. In this thesis, we have built such 'strips' by analysing element connectivity in a recursive manner. For example a two-element strip is the strip of elements that touch the one-element strip and the one-element strip is the strip of elements that touch the interface. The function of this algorithm is shown at the following matlab code.

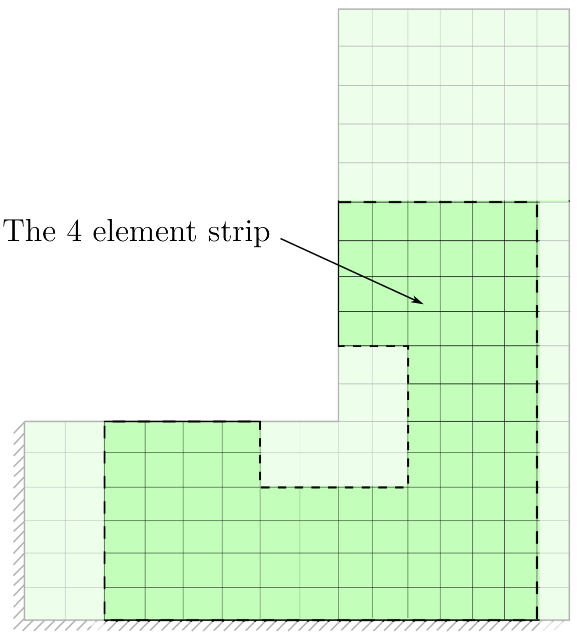


Figure 5.1: A strip of 4 elements of the complement area adjacent to the interface

Matlab Code 5.1: Recursive function get element strips attached

```
function elementStripIDs=getElementStripsAttachedToGlobalElements(self,↩
    numberOfStrips,ElementsIDs)
            if numberOfStrips==1
                [a interfaceNodesID]=self.↩
                    getGlobalInterfaceNodeListForGlobalElements(↩
                    ElementsIDs);
                elementStrip=[];
                for i=1:size(self.globalDomain.elementList,2)
                    nodes=self.globalDomain.getElement(i).nodeList;
                    for j=1:size(nodes,2)
                        if sum(nodes(j).ID==interfaceNodesID)==1 && sum(↩
                            self.globalDomain.getElement(i).ID==↩
                            ElementsIDs)==0
                            elementStrip=[elementStrip self.globalDomain.↩
                                getElement(i).ID];
                            break;
                        end
                    end
                end
                elementStripIDs=elementStrip;
            elseif numberOfStrips>1
                backElementStripIDs=self.↩
                    getElementStripsAttachedToGlobalElements(↩
                    numberOfStrips-1,ElementsIDs);
                ElementsIDs=[ElementsIDs backElementStripIDs];
                elementStripIDs=[backElementStripIDs self.↩
```

```
                    getElementStripsAttachedToGlobalElements(1,ElementsIDs↩
                    )];
                end
            end
```

It is obvious that the more element strips someone chooses the better the approximation is and hence, better results are expected at the convergence of the method.

Matlab Code 5.2: Function to get short scale approximation to Schur complement

```
function Dgamma=getShortScaleApproximationToSchurComplement(self,↩
    numberOfElementStrips)
      model.globalDomain.construct;
            elementStripIDs=model.getElementStripsAttachedToGlobalElements↩
                (numberOfElementStrips,model.globalElementIDs{1});
            [constrainedNodes constrainedNodesIDs]=model.↩
                getGlobalInterfaceNodeListForGlobalElements([model.↩
                globalElementIDs{1} elementStripIDs]);
            ElementStrip=ElementList(model.globalNodeList);
            for i=elementStripIDs
                ElementStrip.addElement(model.globalDomain.getElement(i))
            end
            ElementStrip.construct;
            ElementStrip.nodeList.setPrescribedDisplacements(↩
                constrainedNodes.getDOF,zeros(size(constrainedNodes.getDOF↩
                )));
            interfaceDOF=model.globalInterfaces{1}.getDOF;
            Dgamma=ElementStrip.staticCondensation(interfaceDOF).↩
                getStiffnessMatrix();

            model.globalNodeList.deletePrescribedDisplacements;
        end
```

To summarize, the algorithm for the short scale approximation of the Schur complement is

1. Create a model with the element strip that we have choose

2. Prescribe zero displacements at the other side of the interface of the model

3. Compute the Schur complement of the model at the interface to get the short-scale approximation

$$\widetilde{\mathbf{S}}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}} = \mathbf{K}_{\mathbf{\Gamma\Gamma},\mathbf{C}} - \mathbf{K}_{\mathbf{\Gamma C}}\mathbf{K}_{\mathbf{CC}}{}^{-1}\mathbf{K}_{\mathbf{C\Gamma}} \tag{5.8}$$

where $\widetilde{\mathbf{S}}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}$ is the approximation of the Schur complement of the complement area, $\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}$ and the subscript E denotes the rest of the model except from the interface. The model is shown in figure 5.2.

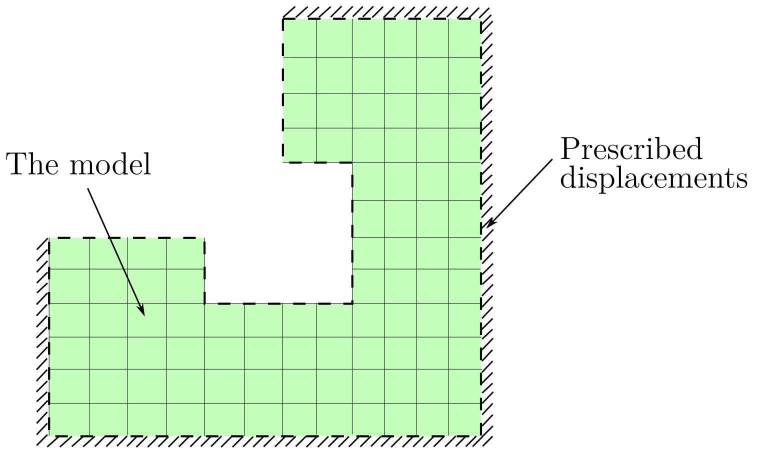


Figure 5.2: The model for the short scale approximation

5.4 Long-scale approximation

The long scale approximations are inspired from homogenization techniques. In these techniques you prescribe a small number of interface loads on the whole subdomain, representing the large scale behaviour. In comparison, short-scale techniques consist in prescribing every possible interface load (i.e. performing condensation) on a small part of the subdomain.

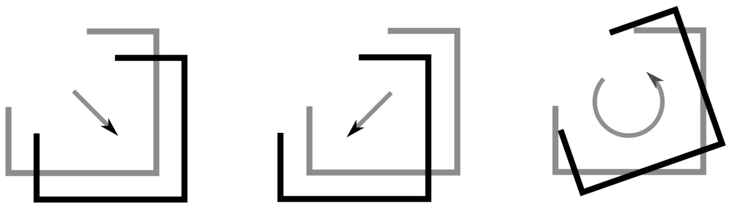
There two ways for this method. We can either prescribe displacements to the interface or we can prescribe loads. If we prescribe displacements we will get an approximation to Schur complement, but if we prescribe loads we will get an approximation to Schur complement's inverse, which is not the approximation that we are interested at. In addition, prescribing loads would require to split the complement area from the area of interest (split global model in two), which is not our case and it would be intrusive. This is why prescribing displacements is chosen. Finally, this choice avoids any potential 'floating subdomain' problem, that could occur with prescribed forces and require a specific algorithm.

In this framework, the 'homogenized', approximate Schur complement is defined as

$$\widetilde{\mathbf{S}}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}} = \mathbf{\Pi}\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}\mathbf{\Pi}^{\intercal} \tag{5.9}$$

where $\mathbf{\Pi}$ is a linear operator projecting interface forces onto some low-dimensional 'macro' space, and its transpose $\mathbf{\Pi}^\intercal$ does the same with interface displacements. This choice guarantees that the approximation is symmetric. Of course, the matrices in the equation above are never computed explicitly, and the result is obtained by submitting the global model to simultaneous load vectors.

The efficiency of the method obviously depends on the choice of these 'macro' spaces. For 'macro' displacements, a common choice is the space of interface rigid body motions (translations and rotations); this allows a convenient definition of 'macro' forces by duality arguments. In this study, we have chosen the space of affine fields as the 'macro' displacement space; this includes translations and rotations, but also stretching and distorting modes, as shown in Figure 5.3. For the sake of simplicity, we always handle the interface as a whole: decomposing it would probably require user intervention in the case of very irregular meshes.


(a) Translations and rotations

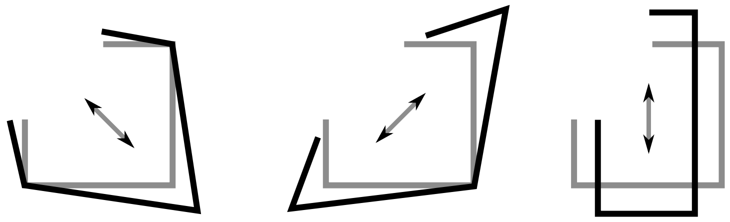

(b) Stretching and distorting modes

Figure 5.3: Affine displacement fields on the interface

Concerning 'macro' forces, two definitions are possible. The simplest choice is to use the same 'macro' space as for the displacements. If $\mathbf{E}$ is a rectangular matrix whose columns form an orthonormal basis of that space (i.e. so that $\mathbf{E}^\intercal\mathbf{E} = \mathbf{I}$), the projector can then be defined as

$$\mathbf{\Pi} = \mathbf{\Pi}^\intercal = \mathbf{E}\mathbf{E}^\intercal \tag{5.10}$$

that is, as an orthogonal projector (represented as a symmetric matrix).

However, that choice may seem too restrictive because a 'macro' space that makes sense for nodal displacements does not necessarily make sense for nodal forces. For that reason, we also considered different 'macro' space and projectors for the two quantities (we require, however, their dimensions to be the same). If $\mathbf{E}$ and $\mathbf{F}$ are the rectangular matrices whose columns respectively define the displacement and force 'macro' bases, and if we assume these bases are biorthonormal (i.e. $\mathbf{E}^\intercal\mathbf{F} = \mathbf{F}^\intercal\mathbf{E} = \mathbf{I}$), then the projectors can be defined as

$$\mathbf{\Pi} = \mathbf{F}\mathbf{E}^\intercal \tag{5.11}$$

$$\mathbf{\Pi}^\intercal = \mathbf{E}\mathbf{F}^\intercal \tag{5.12}$$

Since 'macro' displacements include unit translations and rotations, the corresponding 'macro' forces will have unit resultants and moments. Apart from this, there is no restriction to the choice of the 'macro' force space. For reasons that will be explained later, we have chosen to define this space as the space of reaction forces to prescribed 'macro' displacements. Thus, once these reaction forces have been computed, its basis F is sought as

$$\mathbf{F} = \mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}\mathbf{E}\mathbf{P} \tag{5.13}$$

where $\mathbf{P}$ is a basis transformation matrix; the biorthogonality condition then gives

$$\mathbf{E}^\intercal\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}\mathbf{E}\mathbf{P} = \mathbf{I} \tag{5.14}$$

which means P has to be the inverse or pseudoinverse of $\mathbf{E}^\intercal\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}\mathbf{E}$. Once this pseudoinverse is computed, the 'macro' force basis $\mathbf{F}$ is directly formed by equation (5.13). Those 'macro' basis vectors do not correspond to particularly noticeable force distributions. However, they have the desirable properties of having unit resultants or moments (or higher order moments, due to the stretching and distortion terms), and of containing relevant mechanical information on the subdomain's response.

To summarize, the algorithm for the long scale approximation of the Schur complement is

1. Create matrix $\mathbf{E}$. Each column is a 'macro' displacement and the rows are the degrees of freedom of the interface.
2. Prescribe each 'macro' displacement (each column of $\mathbf{E}$) on the interface Γ of the global model and solve to get the loads of the interface.
3. Calculate matrix $\mathbf{P}$ from equation (5.14). The calculated loads from the preveous step is the matrix $\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}\mathbf{E}$
4. Calculate $\mathbf{F}$ from equation (5.13)

5.5 Two-scale Schur complement approximation

It is has been suggested from Gendre et al. [7] that these suggested approximations, the short-scale and the long-scale, do not contain enough information to give a good representation of the complement area's influence. It seems necessary to represent both the vicinity of the interface and the global behaviour of the subdomain correctly. So, a technique is proposed for combining these approximations, in order to build an efficient representation of a subdomain's stiffness. It is only used to the get an approximation of the Schur complement of the complement area in the case of this thesis, but could certainly have more applications, for example to building preconditioners in domain decomposition methods. [16]

The logic behind this technique can be found in Saint-Venant's principle, which is a well-known empirical principle commonly used in mechanical engineering in general, and in the beam theory in particular. The Saint Venant's principle stays that '*... the strains that can be produced in a body by the application, to a small part of its surface, of a system of forces statically equivalent to zero force and zero couple, are of negligible magnitude at distances which are large compared with the linear dimensions of that part*' [11].

In other words, at large distances from the system of forces, the motion of the 'body' is almost rigid (since strains are negligible). If we could identify this rigid body motion and subtract it from the displacement field, then the residual (non-rigid) displacements would be almost zero. Therefore, clamping that 'body' at a sufficient distance from the system of forces should not introduce a significant error in any component of the solution, see Figure 5.4. This should be true, in particular, for displacements of points where the forces are applied.

Let us apply this principle to the complement domain, subjected to interface forces $\boldsymbol{\lambda}_\Gamma$. Assume that $\mathbf{E}^\intercal \lambda_\Gamma = 0$; since $\mathbf{E}$ contains interface translations and rotations, then $\boldsymbol{\lambda}_\Gamma$'s resultant force and moment are zero. Therefore, the principle above states that the complement domain could be replaced with an element strip such as those presented in Section 5.3, clamped at a sufficient distance from the interface; denoting its Schur complement by $\mathbf{D}_{\boldsymbol{\Gamma}}$, one should then have

$$\mathbf{S}_{\boldsymbol{\Gamma},\mathbf{C}}^{\mathbf{G}\,-1}\boldsymbol{\lambda}_\Gamma \approx \mathbf{D}_{\boldsymbol{\Gamma}}^{-1}\boldsymbol{\lambda}_\Gamma + \mathbf{E}\mathbf{x} \tag{5.15}$$

where $\mathbf{x}$ is a (possibly unknown) amplitude vector, and $\mathbf{E}\mathbf{x}$ represents the rigid body motion that was omitted; $\mathbf{x}$ might be zero if each of the complement area's rigid body modes are blocked. Moreover, since every equation written in the complement area is linear, this rigid body motion has to be

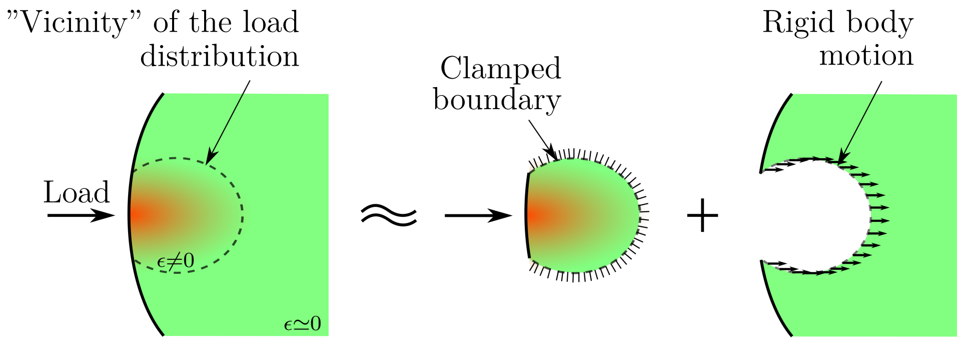


Figure 5.4: Saint Venant’s principle

a linear function of the loading; that is, there exists a rectangular matrix $\mathbf{C}$ such that

$$\mathbf{x} = \mathbf{C}\boldsymbol{\lambda}_\Gamma \tag{5.16}$$

Determining this matrix could be difficult in practice, but luckily it will not be needed in the final result. By combining the two equations above, the following postulate is obtained: for any interface load vector $\boldsymbol{\lambda}_\Gamma$

$$\text{if } \mathbf{E}^\intercal\boldsymbol{\lambda}_\Gamma = 0, \text{ then } {\mathbf{S}^{\mathbf{G}}_{\boldsymbol{\Gamma},\mathbf{C}}}^{-1}\boldsymbol{\lambda}_\Gamma \approx \left[{\mathbf{D}_{\boldsymbol{\Gamma}}}^{-1} + \mathbf{E}\mathbf{C}\right]\boldsymbol{\lambda}_\Gamma \tag{5.17}$$

We can now introduce our two-scale approximation to the Schur complement. It basically consists in splitting, using the projectors defined in Section 5.4, the space of interface forces in two:

1. a ‘micro’ space, design to contain only loads with zero resultant force and moment, thus that can be analyzed on a short-scale approximation, like those in Section 5.3
2. and a ‘macro’ space with a low dimension, that will be analyzed on the whole complement area with homogenization-like techniques, like in Section 5.4

The formulation is a straightforward application of this idea. First, notice that both definitions of the ’macro’ force projector $\boldsymbol{\Pi}$, that are (5.10) and (5.11), imply the following property: for any interface force $\boldsymbol{\lambda}_\Gamma$

$$\mathbf{E}^\intercal\left(\mathbf{I} - \boldsymbol{\Pi}\right)\boldsymbol{\lambda}_\Gamma = 0 \tag{5.18}$$

This means that the resultant force and moment of $\left(\mathbf{I} - \boldsymbol{\Pi}\right)\boldsymbol{\lambda}_\Gamma$ are always zero. Therefore, if we take an arbitrary $\boldsymbol{\lambda}_\Gamma$ and write the following ’macro/micro’ decomposition

$$\boldsymbol{\lambda}_\Gamma = \boldsymbol{\Pi}\boldsymbol{\lambda}_\Gamma + \left(\mathbf{I} - \boldsymbol{\Pi}\right)\boldsymbol{\lambda}_\Gamma \tag{5.19}$$

and left-multiply it by ${\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}}^{-1}$, then the postulate (5.17) gives

$${\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}}^{-1}\boldsymbol{\lambda}_\Gamma \approx {\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}}^{-1}\mathbf{\Pi}\boldsymbol{\lambda}_\Gamma + \left[{\mathbf{D}_\mathbf{\Gamma}}^{-1} + \mathbf{E}\mathbf{C}\right]\left(\mathbf{I} - \mathbf{\Pi}\right)\boldsymbol{\lambda}_\Gamma \tag{5.20}$$

And since this is true for any arbitrary $\boldsymbol{\lambda}$, the following approximation is obtained

$${\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}}^{-1} \approx {\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}}^{-1}\mathbf{\Pi} + \left[{\mathbf{D}_\mathbf{\Gamma}}^{-1} + \mathbf{E}\mathbf{C}\right]\left(\mathbf{I} - \mathbf{\Pi}\right) \tag{5.21}$$

At a first glance, this definition seems to involve the response of the complement domain to prescribed 'macro' forces, due to the ${\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}}^{-1}\mathbf{\Pi}$ term; as explained in the previous section, prescribing interface forces on the complement domain is an intrusive operation. However, this difficulty can be lifted by choosing the 'macro' force space accordingly—that is, by defining it as the space of reaction forces to prescribed 'macro' displacements, as specified by Equations (5.11)-(5.14). With this definition, once these reactions forces have been computed, obtaining ${\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}}^{-1}\mathbf{\Pi}$ is no longer a problem. In addition, Equations (5.11), (5.13) and (5.14) give the following identities

$$\begin{aligned}{\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}}^{-1}\mathbf{\Pi} &= \mathbf{\Pi}^\intercal{\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}}^{-1} \\ &= \mathbf{\Pi}^\intercal{\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}}^{-1}\mathbf{\Pi} \\ &= \mathbf{E}\left[\mathbf{E}^\intercal\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}\mathbf{E}\right]^{-1}\mathbf{E}^\intercal\end{aligned} \tag{5.22}$$

hat is, the response to prescribed 'macro' forces is always a 'macro' displacement, and the response to a prescribed 'macro' displacement is always 'macro' forces. In other terms, the following decomposition holds

$${\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}}^{-1} = \mathbf{\Pi}^\intercal{\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}}^{-1}\mathbf{\Pi} + \left(\mathbf{I} - \mathbf{\Pi}\right)^\intercal{\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}}^{-1}\left(\mathbf{I} - \mathbf{\Pi}\right) \tag{5.23}$$

This decomposition is a direct consequence of the relation between the 'macro' force and displacement spaces, and does not hold with other definitions (i.e. when the two spaces are chosen to be the same).

Another difficulty is that the expression in (5.21) looks unsymmetric, and thus seems to violate a fundamental property of linear elasticity. More precisely, we know that the actual Schur complement is symmetric in our problem; therefore, it is desirable to have a symmetric approximation. However, the identities (5.22) show that the 'macro' term is already symmetric, and only the 'micro' term (with $(\mathbf{I} - \mathbf{\Pi})$) is unsymmetric. Several ways of making it symmetric can be thought of; similar to the decomposition (5.23), we have chosen to slightly adapt Equation (5.21) as follows

$${\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}}^{-1} \approx \mathbf{\Pi}^\intercal{\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}}^{-1}\mathbf{\Pi} + \left(\mathbf{I} - \mathbf{\Pi}\right)^\intercal\left[{\mathbf{D}_\mathbf{\Gamma}}^{-1} + \mathbf{E}\mathbf{C}\right]\left(\mathbf{I} - \mathbf{\Pi}\right) \tag{5.24}$$

Only the 'micro' term changes between the two equations; this change consists in using the short-scale approximation to calculate the 'micro' response to 'micro' forces only. Note that this expression is still a valid consequence of Saint-Venant's principle: when comparing it to (5.23), which is an exact equation, only the response to 'micro' forces is approximated.

Finally, taking the transpose of Equation (5.18) shows that the rigid body term (with $\mathbf{EC}$) vanishes. Thus there is no need to identify rigid body modes and compute the matrix $\mathbf{C}$, and the final, symmetric two-scale Schur complement approximation is

$$\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}{}^{-1} \approx \mathbf{\Pi}^{\intercal}\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}{}^{-1}\mathbf{\Pi} + (\mathbf{I}-\mathbf{\Pi})^{\intercal}\,\mathbf{D}_{\mathbf{\Gamma}}{}^{-1}\,(\mathbf{I}-\mathbf{\Pi}) \tag{5.25}$$

In practice, we are interested by the (primal) Schur complement itself, rather than its inverse. Therefore, the expression above needs to be inverted. Since it involves large full matrices (their size is the number of interface degrees of freedom), a straightforward inversion may be expensive and a cheaper method must be used. For that purpose, Equation (5.25) is rewritten (using the identities (5.22)) as

$$\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}{}^{-1} \approx \mathbf{D}_{\mathbf{\Gamma}}{}^{-1} + \mathbf{\Pi}^{\intercal}\mathbf{\Delta G} + \mathbf{\Delta G\Pi} - \mathbf{\Pi}^{\intercal}\mathbf{\Delta G\Pi} \tag{5.26}$$

where $\mathbf{\Delta G}$ is the difference between the two compliances: that is

$$\mathbf{\Delta G} = \mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}{}^{-1} - \mathbf{D}_{\mathbf{\Gamma}}{}^{-1} \tag{5.27}$$

A direct approximation to $\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}$ is then obtained by applying Woodbury's matrix identity to Equation (5.26); this approximation is inexpensive to compute because this equation is written as $\mathbf{D}_{\mathbf{\Gamma}}{}^{-1}$ plus a small-rank corrective term (namely, its rank is 3 times the dimension of the 'macro' spaces). Using equation (5.13) with equation (5.27) we take the following form

$$\mathbf{\Delta GF} = \mathbf{EP} - \mathbf{D}_{\mathbf{\Gamma}}{}^{-1}\mathbf{F} \tag{5.28}$$

and hence equation (5.26) can take the following form

$$\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}{}^{-1} \approx \mathbf{D}_{\mathbf{\Gamma}}{}^{-1} + \mathbf{UCV} \tag{5.29}$$

where

$$\mathbf{U} = \begin{bmatrix} \mathbf{\Delta GF} & \mathbf{E} & \mathbf{E} \end{bmatrix} \tag{5.30}$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{I} & 0 & 0 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & -\mathbf{F}^{\intercal}\mathbf{\Delta GF} \end{bmatrix} \tag{5.31}$$

$$\mathbf{V} = \begin{bmatrix} \mathbf{E}^{\intercal} \\ (\mathbf{\Delta GF})^{\intercal} \\ \mathbf{E}^{\intercal} \end{bmatrix} \tag{5.32}$$

Then the Woodbury formula gives

$$\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}} = \mathbf{D}_{\mathbf{\Gamma}} - \mathbf{D}_{\mathbf{\Gamma}}\mathbf{U}\mathbf{C}\left[\mathbf{I} + \mathbf{V}\mathbf{D}_{\mathbf{\Gamma}}\mathbf{U}\mathbf{C}\right]^{-1}\mathbf{V}\mathbf{D}_{\mathbf{\Gamma}} \tag{5.33}$$

5.6 Computational procedure

The whole procedure that is used to compute the two scale approximation is summarised in the following algorithm

1. Building of the short-scale approximation
   (a) Extract an 'element strip' from the global model (see Figures 5.1 and 5.2) by analysing element connectivity as explained in Section 5.3
   (b) Perform static condensation on this element strip; this gives the short-scale approximation of the Schur complement, $\mathbf{D}_{\mathbf{\Gamma}}$.
2. Using large-scale technique
   (a) Generate affine displacement fields (see Figure 5.3), compute their values at interface nodes and store them in the rectangular matrix **E**
   (b) Prescribe each of these affine displacement fields on the global model's interface, set other loads to zero, and run a multiple right-hand side analysis to compute the corresponding internal reaction forces on the interface: this gives the product $\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}\mathbf{E}$
   (c) Compute the biorthonormal forces **F** according to Equations (5.13) and (5.14)
3. Computing the two scale approximation
   (a) Compute the two-scale Schur complement approximation by applying Woodbury's matrix identity to Equation (32). All the information needed is contained in **D**, **E** and **F**.
   (b) Use it to the mixed formulation of the method as described in subsection 4.4.2 and in section 5.2.

Matlab Code 5.3: Function to get two scale approximation

```
function Schur=getTwoScaleApproximationToSchurComplement(self,↩
    numberOfElementStrips,numberOfRigidBodyMovements)
```

```
        model=self.multiscaleModel;
        Dgamma=self.getShortScaleApproximationToSchurComplement(↩
            numberOfElementStrips);
        [P F E]=self.getLongScaleApproximationToSchurComplement(↩
            numberOfRigidBodyMovements);

        % Woodbury's matrix identity
        DeltaGF=E*P-Dgamma\F;
        U=[DeltaGF E E];
        C=eye(size(E,2)*3);
        C(size(E,2)*2+1:end,size(E,2)*2+1:end)=-F'*DeltaGF;
        V=[E';DeltaGF' ;E'];
        Schur=Dgamma-Dgamma*U*C/(eye(size(E,2)*3) +V*Dgamma*U*C)*V*↩
            Dgamma;
    end
```

5.7 Computation of the residual

In the previous section the calculation of the stiffness matrix **A** was described. This is used by the iterative method to apply mixed boundary conditions on the local model. However, for the calculation of the residual another stiffness is used, matrix **B**, as we see in equation (4.12)

$$\mathbf{r}_\Gamma = -(\boldsymbol{\lambda}^L_\Gamma + \boldsymbol{\lambda}^G_{\Gamma,C}) + \mathbf{B}(\mathbf{u}^L_\Gamma - \mathbf{u}^G_\Gamma) = [\mathbf{A}+\mathbf{B}](\mathbf{u}^L_\Gamma - \mathbf{u}^G_\Gamma) \tag{5.34}$$

The overall correction is then performed as usual

$$\mathbf{u}^G \leftarrow \mathbf{u}^G + \widetilde{\mathbf{S}}_{\boldsymbol{\Gamma}}^{-1}\mathbf{r}_\Gamma \tag{5.35}$$

where $\widetilde{\mathbf{S}}$ is the operator that is used for the global correction. $\widetilde{\mathbf{S}}_{\boldsymbol{\Gamma}}$ is the Schur complement of the global model (not the complement area)

$$\widetilde{\mathbf{S}}_{\boldsymbol{\Gamma}} = \mathbf{S}^{\mathbf{G}}_{\boldsymbol{\Gamma}} \tag{5.36}$$

A potential drawback is that as shown in the above expression, the correction is made from the latest global solution $\mathbf{u}^G$. However, as we have seen in the previous section, the last local solution, $\mathbf{u}^L$, is usually much better than $\mathbf{u}^G$ when mixed coupling is correctly formulated. In other words, doing the update from $\mathbf{u}^G$ may lead to a strategy that does not converge faster than the approach movement, as shown in Figure 4.6 in Chapter 4, the benefits of the joint connection would and partially lost. The ideal would be that the overall correction takes the form

$$\mathbf{u}^L \leftarrow \mathbf{u}^G + \widetilde{\mathbf{S}}_{\boldsymbol{\Gamma}}^{-1}\mathbf{r}_\Gamma \tag{5.37}$$

As it has already been explained, the optimal choice for $\mathbf{A}$ is the Schur complement of the complement area$\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}$. [2] However we use an approximation of it so we can write

$$\mathbf{A} = \mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}} + \mathbf{\Delta S} \tag{5.38}$$

where $\mathbf{\Delta S}$ is the error of the approximation. Hence equation (5.34) takes the form

$$\mathbf{r}_\Gamma = [\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}} + \mathbf{\Delta S} + \mathbf{B}](\mathbf{u}^L_\Gamma - \mathbf{u}^G_\Gamma) \tag{5.39}$$

and global correction can be written

$$\mathbf{u}^G \leftarrow \widetilde{\mathbf{S}}_{\mathbf{\Gamma}}^{-1}\left(\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}} + \mathbf{B}\right)\mathbf{u}^L + \widetilde{\mathbf{S}}_{\mathbf{\Gamma}}^{-1}\mathbf{\Delta S}\left(\mathbf{u}^L - \mathbf{u}^G\right) + \left[\mathbf{I} - \widetilde{\mathbf{S}}_{\mathbf{\Gamma}}^{-1}\left(\mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}} + \mathbf{B}\right)\right]\mathbf{u}^G \tag{5.40}$$

It is therefore noted that a particularly wise choice is to let

$$\mathbf{B} = \widetilde{\mathbf{S}}_{\mathbf{\Gamma}} - \mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}} \tag{5.41}$$

and so the above equation takes the form

$$\mathbf{u}^G \leftarrow \mathbf{u}^L + \widetilde{\mathbf{S}}_{\mathbf{\Gamma}}^{-1}\mathbf{\Delta S}\left(\mathbf{u}^L - \mathbf{u}^G\right) \tag{5.42}$$

In other words, with the correct choice of $\mathbf{B}$ we have the advantages of the mixed coupling since it starts from the last local solution $\mathbf{u}^L$. Meanwhile, the remaining term on the local solution, appearing in the above expression, is

$$\mathbf{r} = \mathbf{\Delta S}\left(\mathbf{u}^L - \mathbf{u}^G\right) \tag{5.43}$$

However, by introducing the decomposition of $\mathbf{A} = \mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}} + \mathbf{\Delta S}$, the equation of local problem with mixed boundary conditions (5.7) takes the form

$$\mathbf{h}^L_\Gamma(\mathbf{u}^L_\Gamma) + \mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}\mathbf{u}^L + \mathbf{\Delta S}\mathbf{u}^L = \mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}\mathbf{u}^G - \boldsymbol{\lambda}^G_C + \mathbf{\Delta S}\mathbf{u}^L \tag{5.44}$$

or, using the global solution of (5.3)

$$\mathbf{\Delta S}\left(\mathbf{u}^L - \mathbf{u}^G\right) = \mathbf{b}^G_{\Gamma,C} - \mathbf{h}^L_\Gamma(\mathbf{u}^L_\Gamma) - \mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}\mathbf{u}^L \tag{5.45}$$

This situation is shown schematically in Figure 4.6. The second stiffness matrix, $\mathbf{B}$, must be computed as the contribution of the area of interest to the Schur complement of the global area. This means that as $\mathbf{B} = \widetilde{\mathbf{S}}_{\mathbf{\Gamma}} - \mathbf{S}^{\mathbf{G}}_{\mathbf{\Gamma},\mathbf{C}}$ the matrix $\mathbf{B}$ is the Schur complement of the area of interest

$$\mathbf{B} = \mathbf{S}^{\mathbf{L}}_{\mathbf{\Gamma},\mathbf{I}} \tag{5.46}$$

6 Non-conforming interfaces

6.1 Introduction

The biggest advantage of this method is that we can add details or a different constitutive law in an area of an already given model. The local model can be a totally different model in terms of material, type of elements and geometry comparing with the global linear model. In that way new technologies, different elements, new geometries and non-linearities can be easily mixed with old linear models. For this to be done in a totally free way, the method must also be able to handle non-conforming interfaces. In this thesis, a simple and quick way was used to overcome the problem of nonconforming interfaces for situations that add more nodes in the interface of the local model.

6.2 Adding nodes in the local interface

Let the interface of the local model be called local interface and respectively the interface of the global model global interface. Creating a detailed local model requires the addition of nodes in the local interface, making it different from the global interface. In Figure 6.1 we can see the new nodes of the local interface with the red color, while, the other nodes are the same with the global interface.

As in the local interface there are more degrees of freedom in comparison with the global interface there must be a way for these degrees of freedom to be determined. A common technique from domain decompositions methods is the creation of a matrix that connects the displacements of the new degrees of freedom with the displacements of the the ones that already exist. [17] This matrix allows us to determine the displacements of the local interface given the displacements of the global interface. To create this matrix we

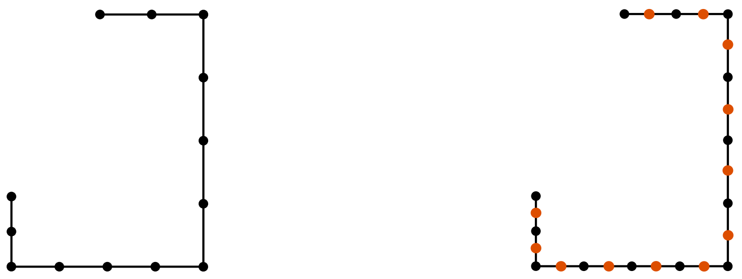

(a) Global interface (b) Local interface

Figure 6.1: Non-conforming interfaces

must firstly determine the relationship between each of the new degrees of freedom with the existing ones. For example, the displacements of a node that is in the middle of two others as shown in the Figure 6.2 is

$$\mathbf{u}_C^L = \frac{1}{2}\mathbf{u}_A^G + \frac{1}{2}\mathbf{u}_B^G \tag{6.1}$$

where $\mathbf{u}_C$ are the displacements of the node C using the displacements of the adjacent nodes A and B, while the displacements of the nodes A and B are the same for the local and the global interface.

$$\mathbf{u}_A^L = \mathbf{u}_A^G \tag{6.2}$$
$$\mathbf{u}_B^L = \mathbf{u}_B^G \tag{6.3}$$

Expressing the above equations in matrix form we get

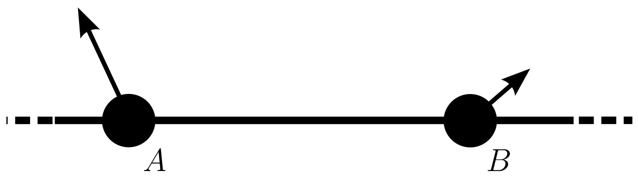
$$\left\{ \begin{array}{c} \mathbf{u}_A \\ \mathbf{u}_B \\ \mathbf{u}_C \end{array} \right\} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \left\{ \begin{array}{c} \mathbf{u}_A \\ \mathbf{u}_B \end{array} \right\} \tag{6.4}$$

In that way, a matrix is created that represents the dependence of the degrees of freedom of the local interface with them of the global interface.

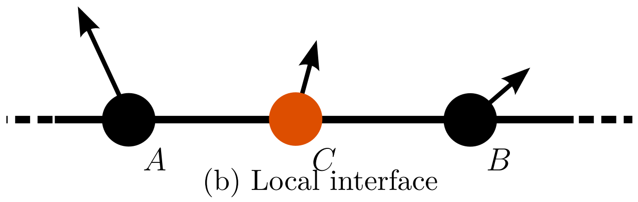
$$\mathbf{u}_\Gamma^L = \mathbf{\Lambda}\mathbf{u}_\Gamma^G \tag{6.5}$$

Now if we want to distribute the forces from the dense local interface to the global interface, we can use the same matrix, $\mathbf{\Lambda}$. To be more specific, we should use its transpose. Matrix $\mathbf{\Lambda}^\intercal$ would distribute the forces from the local interface to the global.

$$\mathbf{f}_\Gamma^G = \mathbf{\Lambda}^\intercal \mathbf{f}_\Gamma^L \tag{6.6}$$



(a) Global interface



(b) Local interface

Figure 6.2: Dependence of the new degrees of freedom

6.3 The new algorithm

The two-scale Schur's complement approximation, is an approximation of the Schur's complement of the global model on the interface. As the global model remains the same so does the Schur's complement approximation. We use the matrix $\mathbf{B}$ to take the local interface displacements from the global ones, and its transpose to take the global interface loads from the local ones. In that way the analysis procedure is:

1. **Local Analysis.** The local analysis is performed in the same way. Because we apply only loads in the local analysis, there is no need to change something with the non-conforming interfaces. The loads are not applied at all the local interface nodes but on the common nodes with the global interface.

2. **Calculation of the residual.** There is a change in the calculation of the residual. For the computation of the residual we use the global and the local interface displacements. For the non-conforming interfaces a change in the global displacements is needed in order to be of the same order with the local displacements. This is done with the use of the matrix $\mathbf{\Lambda}$ as mention before.

$$\mathbf{r}_{\Gamma}^{L} = [\mathbf{A} + \mathbf{B}](\mathbf{u}_{\Gamma}^{L} - \mathbf{\Lambda}\mathbf{u}_{\Gamma}^{G}) \tag{6.7}$$

3. **Global correction.** As the residual loads are computed from the local interface, they must be turned into loads for the global interface using

matrix $\mathbf{\Lambda}^\intercal$.

$$\mathbf{r}_\Gamma^G = \mathbf{\Lambda}^\intercal \mathbf{r}_\Gamma \tag{6.8}$$

Then the method goes on in the same way as the other variations.

This is a quick and easy method to solve some types of non-conforming interfaces. For other type of non-conforming interfaces a mortar element technique can be used in a similar way.

7 Putting theory into practise

7.1 Introduction

The main objective of this thesis was to turn the previous theory into practice and arrive to some conclusions. For that reason a software was needed to be created, which would implement all the necessary methods and algorithms. Matlab was chosen as the preferred code, where everything was written from scratch. No ready finite element programs were used. Matlab was chosen for the following reasons:

1. It is a powerful tool for processing matrices (in finite elements almost all algorithms require the handling of matrices and vectors). Matlab has many and powerful built-in functions for the handling of matrices that reduces the computational cost and makes the program shorter, more efficient and easier to comprehend.

2. It is simple and user friendly, but also has a powerful IDE (integrated development environment), which allows a quick and easy implementation of methods. In addition, it has strong tools for profiling and testing the algorithms.

3. It was the main program-mathematical tool being educated in the faculty of civil engineers of NTUA.

The software was chosen to be written in an object-oriented way. The efficiency of object-oriented programming in computational problems is a highly debated topic, especially now with the raising interest in parallel computing. The main advantages of object-oriented programming are :

1. It provides a clear modular structure for programs which makes it good for defining abstract datatypes where implementation details are hidden and the unit has a clearly defined interface. In that way, the final

algorithm is easy to understand even from people not familiar with programming.

2. It makes it easy to maintain and modify existing code as new objects can be created with small differences to existing ones. This leads to maintainable and reusable code that is easily adapted to new problems and changes.

3. It provides a good framework for code libraries where supplied software components can be easily adapted and modified by the programmer.

Matlab provides the ability to build object-oriented programs, but in action this turned to have some disadvantages. The major disadvantage was that while Matlab was extremely quick and efficient with matrix operations it turned to be slower with object classes, especially when they had the 'handle' attribute which is essential for object-oriented programming. Hence, this leads sometimes in misleading results when we have to compare methods that have matrices operations with methods that have extended handling of objects. Furthermore, it seems that using classic object-oriented programming classes (such as the ones used in Java or C++ programming) is not the best approach for a Matlab object-orienting program. The best and maybe the only good choice is to write vectorized algorithms. Last but not least, as Matlab does not use pointers in the handling of matrices there is a significant drawback for finite element programs that their objects require big amounts of matrices exchanges. However, Matlab remains a powerful tool for creating such programs with a plethora of mathematical and engineering libraries.

7.2 Creating linear Finite Element software

Firstly, a classic linear finite element program was needed. We started by creating the classes of 'Node' and 'Element'. The logic of the algorithm is:

1. A node list is created with all the nodes that will be used from the domain. The 'Node' object has as properties all the data that are required as is shown in the below Matlab Code.

Matlab Code 7.1: Properties of 'Node' object

```
classdef Node < handle
    properties
        ID
        x
        y
        z

        releases=[0 0 0 0 0 0]
        constraints=[0 0 0 0 0 0]
        degreesOfFreedom=[0 0 0 0 0 0]
        dof=[]

        loads=[0 0 0 0 0 0]
        prescribedDisplacements=[NaN NaN NaN NaN NaN NaN]

        displacements=[0 0 0 0 0 0]
        stresses=[]

        stressesFromElements=[]
        elementsConnected=[]
    end
```

The creation of a node requires its ID and its coordinates. Loads, prescribed displacements and constraints are setted directly to the node using functions of the 'Node' object.

2. Once the node list has been created, we create an element list. The function that creates an element has as input the nodes that the element has, which must be in correct order, and the material of the element. We create a different 'Element' class for each type of finite element. The main functions of an element object is the calculation of its stiffness matrix and the calculation of its stresses and loads given the nodes displacements. As an example the calculation of the stiffness matrix of the linear quadrilateral element is shown below:

Matlab Code 7.2: Function getStiffnessMatrix of LinearQuadrilateral class

```
function k=getStiffnessMatrix(self)
            k=zeros(8);
            C=self.material.C;
            n=size(self.gaussPointList,2);
            xy=self.nodesXY;
            gaussPoint=self.gaussPointList;
            for i=1:n
                h=gaussPoint(i).x;
                j=gaussPoint(i).y;
                J=1/4*[-(1-h) (1-h) (1+h) -(1+h);
                       -(1-j) -(1+j) (1+j) (1-j)]*xy;
                B1=1/det(J)*[J(2,2) -J(1,2) 0 0;
                        0 0 -J(2,1) J(1,1);
                        -J(2,1) J(1,1) J(2,2) -J(1,2)];
                B2=1/4*[-(1-h) 0 (1-h) 0 (1+h) 0 -(1+h) 0;
                        -(1-j) 0 -(1+j) 0 (1+j) 0 (1-j) 0;
                        0 -(1-h) 0 (1-h) 0 (1+h) 0 -(1+h);
                        0 -(1-j) 0 -(1+j) 0 (1+j) 0 (1-j)];
                B=B1*B2;
                kgauss=B'*C*B*det(J)*self.thickness;
                k=k+kgauss;
            end
        end
```

Furthermore, the element object sets the releases of the nodes that it has. In that way, we can have different types of elements with different releases in the same model (the node that connects different types of elements has the releases from both types).

3. The next step is the creation of the stiffness matrix. Once the elements have determined the releases of each node, the node list can now determine the degrees of freedom of each node. The domain stiffness matrix is created by adding the stiffness matrix of each element to the correct position according to the degrees of freedom of each element. For this procedure a class has been created that represents the stiffness matrix. This 'StiffnessMatrix' object has as properties the stiffness matrix, that can be saved in more condensed way (sparse matrix, symmetric sparse matrix) instead of full matrix, and its degrees of freedom as a vector. The stiffness matrix class has the following functions:

   (a) add element's stiffness matrix

   (b) delete degrees of freedom

   (c) get matrix of specific degrees of freedom

   (d) make static condensation at degrees of freedom

4. After the creation of the stiffness matrix, we can proceed to the computation of the displacements by solving the known equation

$$\mathbf{K} * \mathbf{d} = \mathbf{P} \tag{7.1}$$

   where **K** is the stiffness matrix, **d** is the displacement vector and **P** is the load vector. The load vector is created from the node list by running throughout the nodes to find the applied loads. In a similar way the prescribed displacements are found and their degrees of freedom. The procedure is shown in the following matlab code:

Matlab Code 7.3: Function that solves the stiffness linear equation

```
function [displacementVector forcesVector]=classicSolver(↩
    StiffnessMatrix,prescribedDisplacements,↩
    prescribedDisplacementsDOF,loadVector)
            % Displacements
            K=StiffnessMatrix;
            cDOF=prescribedDisplacementsDOF;
            if isempty(cDOF)
                D=rcond(full(K.getStiffnessMatrix));
                if D<10e-5
                    warning('Probably Floating domain');
                end
                displacementVector=K.getStiffnessMatrix\(loadVector)↩
                    ;
                forcesVector=-loadVector;
            else
                prescribedDisp=prescribedDisplacements;
                Ke=K.deleteDOF(cDOF);
                eDOF=Ke.getDOF;
                Kd=K.getKce(eDOF,cDOF);
                Re=loadVector;
                Re(cDOF)=[];
                de=Ke.getStiffnessMatrix\(Re-Kd.getStiffnessMatrix*↩
                    prescribedDisp);

                %Reaction forces
                Kdd=K.getKcc(cDOF);
                Kde=K.getKce(cDOF,eDOF);
                Rd=Kde.getStiffnessMatrix*de+Kdd.getStiffnessMatrix*↩
                    prescribedDisp;

                % Displacement vector
                disp=zeros(size(StiffnessMatrix.getDOF,2),1);
                disp(eDOF)=de;
                disp(cDOF)=prescribedDisp;
                displacementVector=disp;

                % Forces vector
                R=loadVector;
                R(cDOF)=Rd-R(cDOF);
                forcesVector=R;
            end
end
```

5. The displacements are saved in the correct node's properties and then the elements can calculate the loads, the stresses and everything else it is needed for the post-procession. Functions that take as input the element list can now plot the deformed model or the stresses.

7.3 Creating the non-linear Finite Element software

The non-linear finite element is the next object that must be created. For this reason we build firstly an non linear material class using the theory and the functions that are presented in Chapter 2.

The 'NonLinearMaterial' class represents the non-linear material object. To create this object, the Young modulus, the Poisson ratio and the yield stress are required. The functions that this class has, are the backward Euler return and the construction of the consistent modular matrix. There are also specific functions for the plane stress finite elements.

The non linear element object follows the next in order to use correctly the 'NonLinearMaterial' object.

1. In the construction of the stiffness matrix instead of using the classic modular matrix, the non-linear element receives from the 'NonLinearMaterial' object the consistent tangent modular matrix. The function of the creation of this modular matrix takes as input the stresses of the point where the matrix is calculated (Gauss points in our case).Consequently, the 'NonLinearElement' object has to calculate the stresses of its Gauss points before getting the tangent modular matrix. If this is the first call then the stresses are zero.

2. In each iteration, once the equation (7.1) has been solved, the 'NonLinearElement' object gets the displacements from its nodes. Then, using shape functions, it calculates the new strains in the Gauss points. Having also the preveous strains, it calculates the strain increment of this iteration.

3. Using the previous stresses and the strain increment, a backward Euler return (it is a function of the 'NonLinearMaterial' object) is performed in each Gauss point. As already mentioned the backward Euler return is better than the forward Euler technique. From the backward Euler return the 'NonLinearElement' gets and saves the new stresses at each Gauss point.

4. Using the stresses of each Gauss point, the internal loads of the 'NonLinearElement' are calculated. These loads are used to check the convergence of the non-linear analysis method that is used (Newton-Raphson, arc-length etc.). If there is no convergence then the next iteration starts from step 1 with these new stresses.

The calculation of the stresses by a non-linear quadrilateral element are shown in the following matlab codes:

Matlab Code 7.4: Function that calculates the strains of the NonLinear-Quadrilateral element at specific coordinates

```
function strains=getStrainsFromCurrentNodeDisplacements(self,h,j)
            disp=zeros(1,8);
            xy=self.nodesXY;
            for node=1:4
                disp(node*2-1:node*2)=self.nodeList(node).displacements↩
                    (1:2);
            end
            J=1/4*[-(1-h) (1-h) (1+h) -(1+h);
                -(1-j) -(1+j) (1+j) (1-j)]*xy;
            B1=1/det(J)*[J(2,2) -J(1,2) 0 0;
                    0 0 -J(2,1) J(1,1);
                    -J(2,1) J(1,1) J(2,2) -J(1,2)];
            B2=1/4*[-(1-h) 0 (1-h) 0 (1+h) 0 -(1+h) 0;
                        -(1-j) 0 -(1+j) 0 (1+j) 0 (1-j) 0;
                        0 -(1-h) 0 (1-h) 0 (1+h) 0 -(1+h);
                        0 -(1-j) 0 -(1+j) 0 (1+j) 0 (1-j)];
            B=B1*B2;
            strains=B*disp';
        end
```

Matlab Code 7.5: Function that calculates the new stresses and loads of the NonLinearQuadrilateral element

```
function setStressesToGaussPoints(self)
            n=size(self.gaussPointList,2);
            R=zeros(8,1);
            disp=zeros(1,8);
            for node=1:4
                    disp(node*2-1:node*2)=self.nodeList(node).↩
                        displacements(1:2);
            end
            xy=self.nodesXY;
            for i=1:n
                gaussPoint=self.gaussPointList(i);
                gaussPointStrains=gaussPoint.getStrains;
                h=gaussPoint.x;
                j=gaussPoint.y;
                J=1/4*[-(1-h) (1-h) (1+h) -(1+h);
                    -(1-j) -(1+j) (1+j) (1-j)]*xy;
                B1=1/det(J)*[J(2,2) -J(1,2) 0 0;
                        0 0 -J(2,1) J(1,1);
                        -J(2,1) J(1,1) J(2,2) -J(1,2)];
                B2=1/4*[-(1-h) 0 (1-h) 0 (1+h) 0 -(1+h) 0;
                        -(1-j) 0 -(1+j) 0 (1+j) 0 (1-j) 0;
                        0 -(1-h) 0 (1-h) 0 (1+h) 0 -(1+h);
                        0 -(1-j) 0 -(1+j) 0 (1+j) 0 (1-j)];
                B=B1*B2;
                newGaussPointStrains=B*disp';
                strainIncrement=newGaussPointStrains-gaussPointStrains;
                currentStresses=gaussPoint.getStresses;

                [newStresses isNonlinear]=self.nonLinearMaterial.↩
                    backwardEulerReturn(currentStresses,strainIncrement);
                if isNonlinear
                    gaussPoint.isNonlinear=true;
                end
                R=R+(B')*newStresses*det(J)*self.thickness;
                self.internalLoads=R;
                gaussPoint.setStresses(newStresses);
                gaussPoint.setStrains(newGaussPointStrains);

            end
        end
```

For the non-linear finite element code to be complete, the analyses methods that are described in Chapter 3 are required. Using the load control method (Section 3.2) or the displacement control method (Section 3.3) with the above objects and functions we get a usable non-linear finite element code.

7.4 Analysis with the two-Scale approximation

The main functions for the creation of the two-scale Schur complement's approximation have already been presented in Chapter 5. But in this thesis the code was designed do the whole procedure by itself, without the intervention of the user. In other words, someone has only to submit a linear model and a non-linear material and leave all the rest calculations for the software. The software has to determine the region where the plasticity occurs and then create a local model for that region. Then the two scale analysis is performed with the local and the global model. The algorithm of this multiscale analysis is:

1. First linear analysis. A linear elastic analysis is performed to the global model and the stresses of all the Gauss points are calculated. Then the Gauss points that have crossed the yield surface are found (see Figure 8.3 in the Chapter 8).

2. Creation of the local model. From the Gauss points that have crossed the yield surface, we can now determine which elements determine the non-linear area. It is important to point out here that this area is most likely to be expanded when the elastoplastic properties will be used. This is why the local model should contain these elements as well as one or two element strips around them (see Figure 8.4 in the Chapter 8). Of course the local model is a new non-linear model that is created from the properties of the elements chosen. The global model remains untouched.

3. Computation of the Schur approximation. The short scale approximation is computed (as described in section 5.3), then the long scale (as described in section 5.4) and then the two-scale approximation. These functions have already be described in Chapter 5. If we want to calculate the Schur complement explicitly, in this step, after we have created the local model, we remove from the global model the elements that are in the plastic area (this is an intrusive operation) and then we make

a static condensation on the degrees of freedom of the interface (also intrusive as requires the stiffness matrix).

4. Multiscale iterations. Using the Schur complement the loads of the local model are computed and a non-linear load control analysis is performed as descused in section 3.2. The residual is calculated and it is applied to the global model. These iterations stop when the residual is small enough. Apart from the residual two other error estimators are used. They check the difference between the displacements of the local model and the global model in the interface. The first, n_u^R, uses the displacements of the reference solution while the other, n_u^G, of the gloabl model's displacements.

$$n_u^R = \frac{\|u_\Gamma^L - u_\Gamma^R\|}{\|u_\Gamma^R\|} \tag{7.2}$$

$$n_u^G = \frac{\|u_\Gamma^L - u_\Gamma^G\|}{\|u_\Gamma^G\|} \tag{7.3}$$

Where u_Γ^L, u_Γ^R and u_Γ^G are the displacements of the interface of the local, the reference and the global solution respectively. Many times a good reference solution will not exist, so the residual and the n_u^G are the only indicators that are always available. Here is the matlab code for the multiscale iterations:

Matlab Code 7.6: Function of the multiscale iterations

```
function residualN=multiscaleIteration(self,Schur,B,↩
    globalPrescribedDisplacements,prescribedDislpacementsDOF,↩
    globalLoads)
            GlobalDomain=self.multiscaleModel.globalDomain;
            FinalDisp=GlobalDomain.displacements;
            globalInterface=self.multiscaleModel.globalInterface;
            LocalDomain=self.multiscaleModel.localDomain;
            localInterface=self.multiscaleModel.localInterface;

            elementStrip1=self.multiscaleModel.↩
                getElementStripsAttachedToGlobalElements(1,self.↩
                multiscaleModel.globalElementID);

            i=0;
            residualNorm=1;
            noi=1;


            globalInterfaceDisplacements=GlobalDomain.displacements(↩
                globalInterface.getDOF);
            loads=GlobalDomain.↩
                getLoadsFromNodeDisplacementsForElements(↩
                elementStrip1);
```

```
        globalInterfaceLoads=loads(globalInterface.getDOF);

        while residualNorm>1e-4
            i=i+1;

            % Local Domain solve paper Gendre et al. (11)
            Forces=Schur*globalInterfaceDisplacements-↩
                globalInterfaceLoads;
            %LocalNodes.setLoads(localLoadsDOF,localLoads);
            Loads=LocalDomain.nodeList.getLoads;
            Loads(localInterface.getDOF)=Loads(localInterface.↩
                getDOF)+Forces';
            noi=Solver.↩
                nonLinearLoadControlSolverWithStiffnessAddition(↩
                LocalDomain,localInterface.getDOF,Schur,Loads↩
                ',1);

            %loads=LocalDomain.getLoadsFromStresses;
            %localInterfaceLoads=loads(localInterface.getDOF);
            localInterfaceDisplacements=LocalDomain.↩
                displacements(localInterface.getDOF);

            %Residual
            %residualForces=-(globalInterfaceLoads+↩
                localInterfaceLoads);
            residualDisplacements=localInterfaceDisplacements-↩
                globalInterfaceDisplacements;
            residual=(Schur+B)*residualDisplacements;


            Loads=zeros(size(globalLoads));
            Loads(globalInterface.getDOF)=Loads(globalInterface.↩
                getDOF)+residual;
            %GlobalNodes.setPrescribedDisplacements(↩
                globalInterface.getDOF,residualDisplacements);

            [ddisp loads]=Solver.classicSolver(self.globalK,↩
                globalPrescribedDisplacements,↩
                prescribedDislpacementsDOF,Loads);
            FinalDisp=FinalDisp+ddisp;
            GlobalDomain.setDisplacements(FinalDisp);
            globalInterfaceDisplacements=GlobalDomain.↩
                displacements(globalInterface.getDOF);
            loads=GlobalDomain.↩
                getLoadsFromNodeDisplacementsForElements(↩
                elementStrip1);
            globalInterfaceLoads=loads(globalInterface.getDOF);

            %Residual
            %residualForces=-(globalInterfaceLoads+↩
                localInterfaceLoads);
            residualDisplacements=localInterfaceDisplacements-↩
                globalInterfaceDisplacements;
            residual=(Schur+B)*residualDisplacements;
            residualNorm=norm(residual,2)/norm(globalLoads,2);

            residualN(i,:)=[noi residualNorm];
        end
    end
```

5. Once the iterations have completed and the local and global model are in equilibrium, we check if there are elements in the global model outside the area of interest (the area that has also the local model) that have crossed the yield surface. If there is no such an element then the problem is solved. If there are elements that have crossed the yield surface and are not in the local model, then a new local model is created and the multiscale iterations must be executed again. Of course, we can perform the above check inside the multiscale iteration, which would be more efficient as we would avoid extra iterations.

7.5 Analysis with non-conforming interfaces

The previous algorithm was adapted so that it can handle also non-conforming interfaces. The key matrix for the analysis is the **Λ** matrix that was described in section 6.2. The creation of this matrix is carried out together with the creation of the local model, which is in a finer scale. The local model that is created has four quadrilateral elements in the place where the global model has one. The following matlab code Shows the function that creates the local model and the **Λ** matrix (matrix B in the code):

Matlab Code 7.7: Function create finer model

```
function [localDomain localInterfaceNodeList interfaceNodesID B]=↩
    createFineNonLinearDomain(self,globalElementIDs,NonLinearMaterial)

            [localInterfaceNodeList interfaceNodesID]=self.↩
                getLocalInterfaceNodeListForGlobalElements(↩
                globalElementIDs);
            localDomain=ElementList(self.localNodeList);
            B=eye(localInterfaceNodeList.size*2);
            for i=globalElementIDs
                coarseElement=self.globalDomain.getElement(i);
                ID{1}=coarseElement.nodeList(1).ID;
                ID{2}=coarseElement.nodeList(2).ID;
                ID{3}=coarseElement.nodeList(3).ID;
                ID{4}=coarseElement.nodeList(4).ID;
                ID{5}=ID{1};
                for i=1:4
                x=(self.globalNodeList.getNode(ID{i}).x +
                self.globalNodeList.getNode(ID{i+1}).x)/2;
                y=(self.globalNodeList.getNode(ID{i}).y +
                self.globalNodeList.getNode(ID{i+1}).y)/2;
                IDnew=self.localNodeList.findNode(x, y);
                if IDnew==0
                    self.localNodeList.newNode(x,y);
                    IDnew=self.localNodeList.size;
                    if  not(isempty(find(interfaceNodesID==ID{i}, 1))) && ↩
                        not(isempty(find(interfaceNodesID==ID{i+1}, 1)))
                        localInterfaceNodeList.addNode(self.localNodeList.↩
                            getNode(IDnew));
                        %interfaceNodesID=[interfaceNodesID IDnew];
                        B(size(B,1)+1,2*localInterfaceNodeList.↩
                            findIndexOfID(ID{i})-1)=0.5;
                        B(size(B,1),2*localInterfaceNodeList.findIndexOfID↩
                            (ID{i+1})-1)=0.5;
                        B(size(B,1)+1,2*localInterfaceNodeList.↩
                            findIndexOfID(ID{i}))=0.5;
                        B(size(B,1),2*localInterfaceNodeList.findIndexOfID↩
                            (ID{i+1}))=0.5;
                    end
                end
                IDn{i}=IDnew;
                end
                x=(self.globalNodeList.getNode(ID{1}).x +
                self.globalNodeList.getNode(ID{2}).x +
                self.globalNodeList.getNode(ID{3}).x +
                self.globalNodeList.getNode(ID{4}).x)/4;
                y=(self.globalNodeList.getNode(ID{1}).y +
                self.globalNodeList.getNode(ID{2}).y +
                self.globalNodeList.getNode(ID{3}).y +
                self.globalNodeList.getNode(ID{4}).y)/4;
                self.localNodeList.newNode(x,y);
                IDnew=self.localNodeList.size;
                IDn{5}=IDnew;
                thick=coarseElement.thickness;
                localDomain.addElement(NonLinearQuadrilateral(self.↩
                    localNodeList.getNode(ID{1}),self.localNodeList.↩
                    getNode(IDn{1}),self.localNodeList.getNode(IDn{5}),↩
                    self.localNodeList.getNode(IDn{4}),thick,↩
                    NonLinearMaterial));
                localDomain.addElement(NonLinearQuadrilateral(self.↩
                    localNodeList.getNode(IDn{1}),self.localNodeList.↩
                    getNode(ID{2}),self.localNodeList.getNode(IDn{2}),self↩
                    .localNodeList.getNode(IDn{5}),thick,NonLinearMaterial↩
                    ));
                localDomain.addElement(NonLinearQuadrilateral(self.↩
                    localNodeList.getNode(IDn{5}),self.localNodeList.↩
                    getNode(IDn{2}),self.localNodeList.getNode(ID{3}),self↩
                    .localNodeList.getNode(IDn{3}),thick,NonLinearMaterial↩
                    ));
                localDomain.addElement(NonLinearQuadrilateral(self.↩
                    localNodeList.getNode(IDn{4}),self.localNodeList.↩
                    getNode(IDn{5}),self.localNodeList.getNode(IDn{3}),↩
                    self.localNodeList.getNode(ID{4}),thick,↩
                    NonLinearMaterial));
            end
        end
```

Once the $\mathbf{\Lambda}$ matrix has been computed and the local model has been created, the non-conforming interfaces can be solved using the algorithm descried in section 6.3. If the interfaces are conforming then the $\mathbf{\Lambda}$ matrix is an identity matrix in the size of the interface degrees of freedom. The residual is now computed in this way:

Matlab Code 7.8: Computation of residual

```
residualDisplacements= localInterfaceDisplacements −B1*↩
    globalInterfaceDisplacements;
residual=B*residualDisplacements;
residualNorm=norm(B1'*residual,2)/norm(globalLoads,2);
Loads=zeros(size(globalLoads));
Loads(globalInterface.getDOF)=Loads(globalInterface.getDOF)+B1'*↩
    residual;
```

8 Example 1

8.1 Description of the model

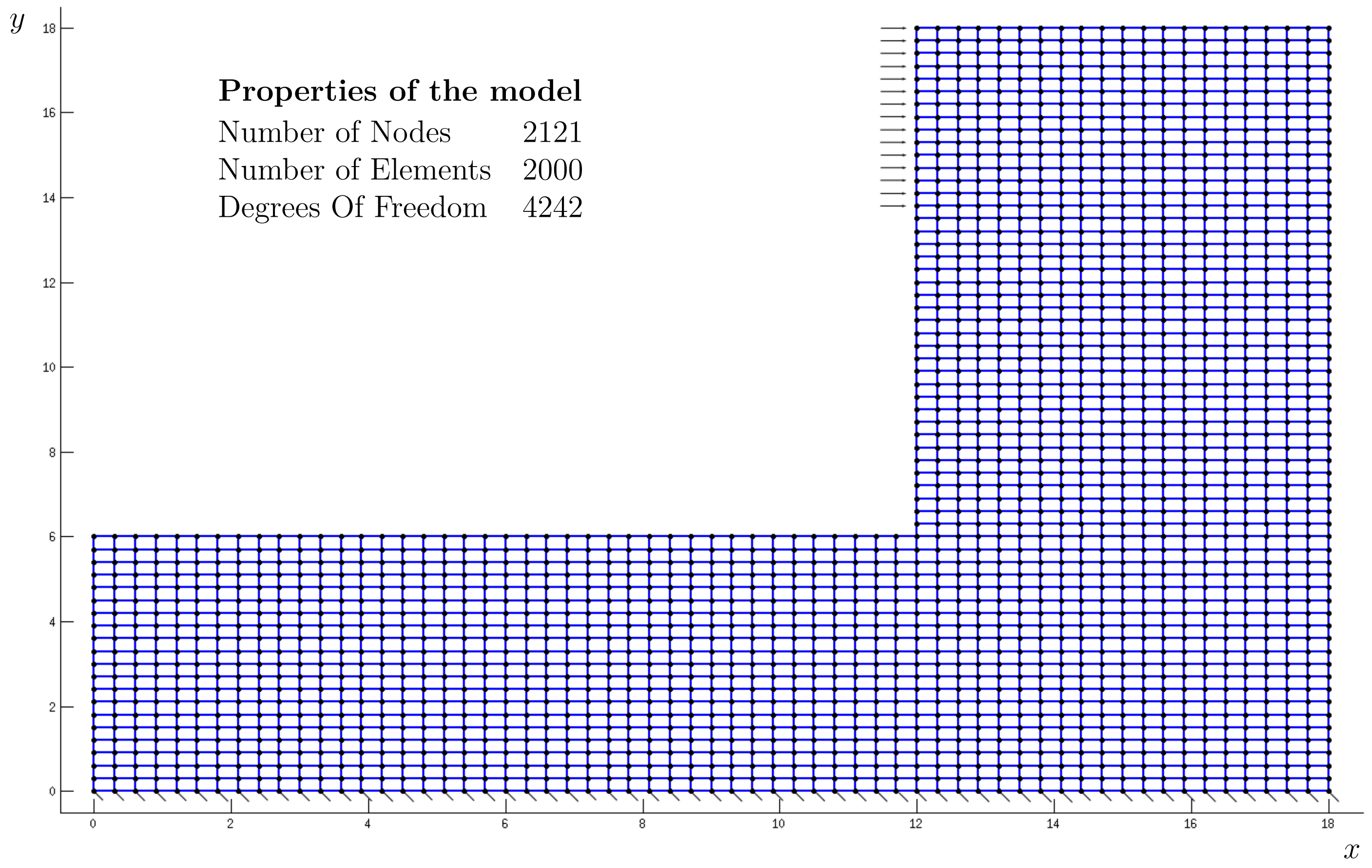
The model is a 2D model, created with linear quadrilateral elements (Figure 8.1). It is a common L-shaped model with regular mesh and a linear constitutive law. The model is fixed on the bottom and the loading is applied on the top, as shown in Figure 8.1. The material was chosen to be steel. There is no physical representation for this model, but it is a good theoretical example in order to apply the methods of the current thesis. Table 8.1 shows the properties of the model.

Young Modulus	$210^6 (KN/m^2)$
Poison Ratio	0.3
Yield Stress	$500000 (KN/m^2)$
Finite Element Thickness	0.1
Number of Nodes	2121
Number of Elements	2000
Total Degrees of Freedom	4242
Free Degrees of Freedom	4182

Table 8.1: Properties of the model in example 1

8.2 Solving the reference problem

First we solve the same problem but with non linear quadrilateral elements, instead of linear. The non linear elements use the von Mises criterion as described in Chapter 2. This solution is considered to be the reference solution,



Properties of the model	
Number of Nodes	2121
Number of Elements	2000
Degrees Of Freedom	4242

Figure 8.1: The model 1

Degrees of freedom	4242
Number of iterations (1-step)	11
Time for calculations (1-step)	88 sec
Number of iterations (10-steps)	38
Time for calculations (10-steps)	329 sec
Gauss points crossed yield surface	45
Elements used non linear law	15

Table 8.2: Results of the reference solution

to which we will compare all the other results. This full non linear model was solved in one loading step, which is used for the time comparison as the other methods also take the load in one step, as well as in 10 loading steps, which is used for the comparison of the error of the other methods as this solution is more accurate. Some properties of this solution are shown in Table 8.2, while Figure 8.2 shows the Gauss points and elements that crossed the yield surface.

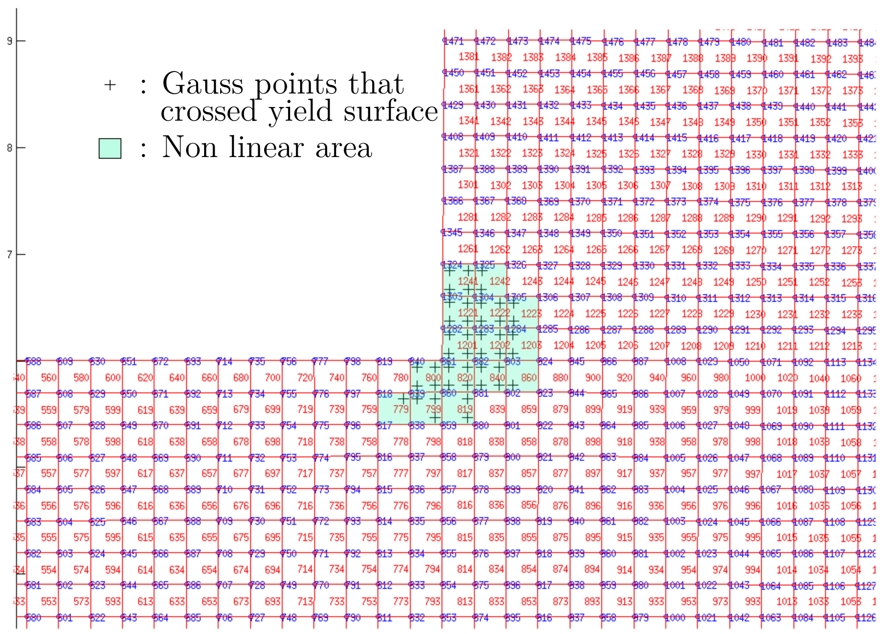


Figure 8.2: Non linear area of the reference problem

8.3 Analysis procedure

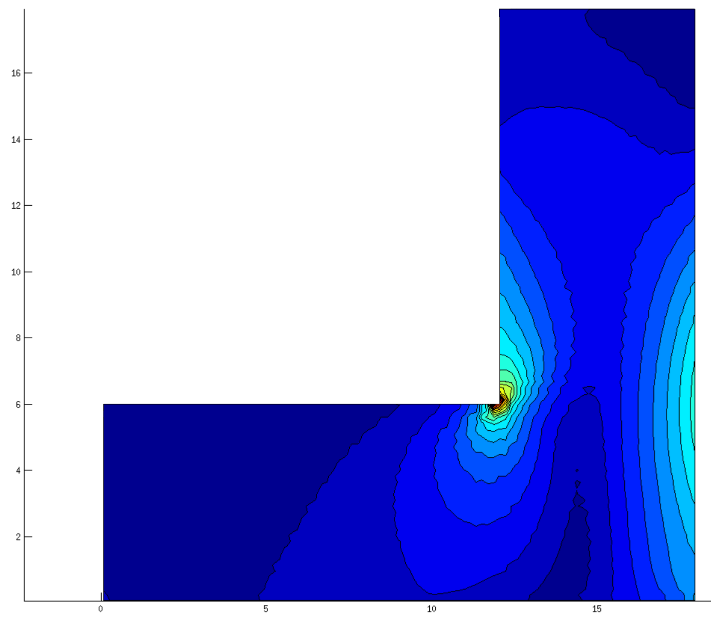
The procedure starts with a linear elastic analysis of the global model. Then the stresses are computed to the Gauss points of every element (Figure 8.3). Using the von Mises criterion the program finds which Gauss points have crossed the yield surface. Then the local model is created. The local model has the elements that their Gauss points crossed the yield surface, as well as a few more strips of elements around them to 'catch' the possible expansion of the non-linear area after a few iterations (see Figure 8.4). By comparing Figure 8.4 and Figure 8.2 we can see that the elements found from the linear analysis are not the same with the elements that finally are in the non linear area. That is way the addition of element strips to the first local model is necessary. We can see that in this case an one-element strip would be enough for covering all the non linear area, but no strip addition would finally lead us to rebuilt the local model in order to expand it. This of course would be computationally costly and should be avoided. Once the local model has been created, the algorithm for the coupling of the two models is used as it was described in Section 4.

8.4 Comparison of the displacement and the mixed variation

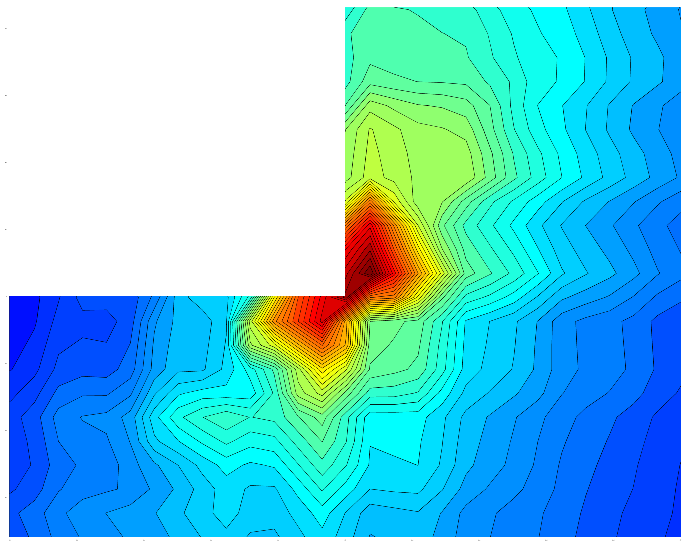
Firstly, we present a comparison of the convergence rate of the displacement and the mixed variation. In Figure 8.5 we can see the error of the displacement and the mixed variation of Section 4.4 in a logarithmic scale for the first four iterations between the two models. The blue curve represents the displacement variation, while the red curve the mixed boundary conditions variation. As we can see the mixed boundary conditions bring significant improvements in convergence rate.

8.5 A research at the parameters of the two-scale approximation

Many analyses were performed with different parameters in order to test the effectiveness of the methods as well as their functionality and efficiency. It was chosen that a two-element strip will be added to the elements that crossed the yield surface and hence, the second local model of the Figure 8.4 was created. First, an analysis with the Schur complement being computed



(a) The von Mises stresses of the model



(b) Zoom at the corner of the model

Figure 8.3: The von Mises stresses after the first linear analysis

explicitly in an intrusive way was occurred. As expected, the coupling was finished after the first local analysis of the model.

Then the analyses that used the Schur complement approximation were performed. The main parameters of the Schur complement approximation

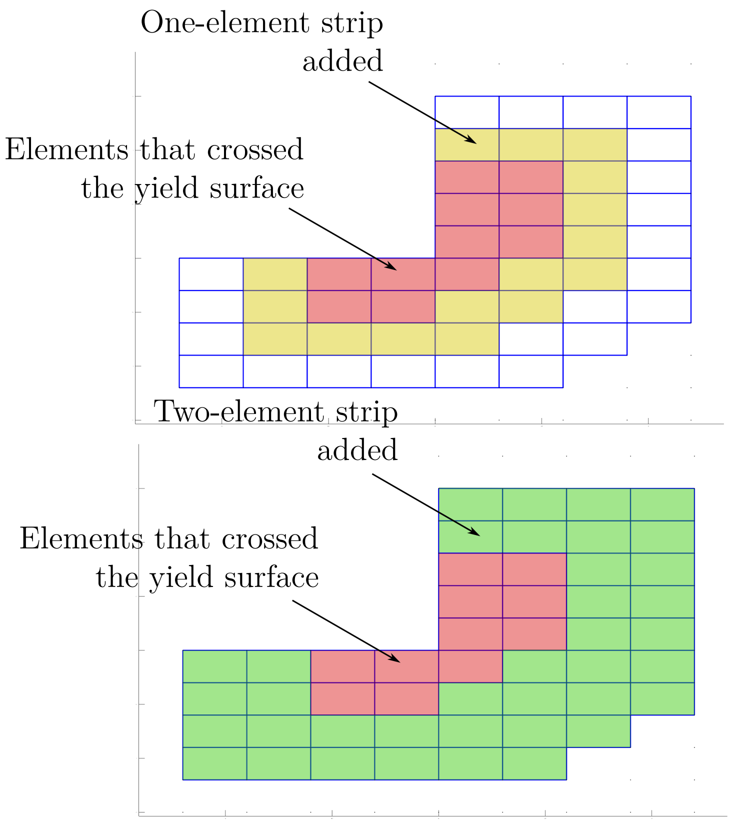

One-element strip added
Elements that crossed the yield surface
Two-element strip added
Elements that crossed the yield surface

Figure 8.4: Creation of the local model

are:

1. The number of the element strips that are used for the short scale approximation (see Section 5.3).

2. The number of the rigid body movements that are used for the long scale approximation (see Section 5.4).

In Table 8.3 we can see the iterations of the local model and the error compared to the reference solution when we change the element strip for the short scale approximation (Figure 8.6). Each row represents a coupling iteration between the local and the global models. As we can see all four analyses ended after seven coupling iterations. Each row shows how many iterations where occurred in each local non linear analysis and the error after applying the residual to the global model. A visualisation of the above data is given in Figures 8.7 and 8.8.

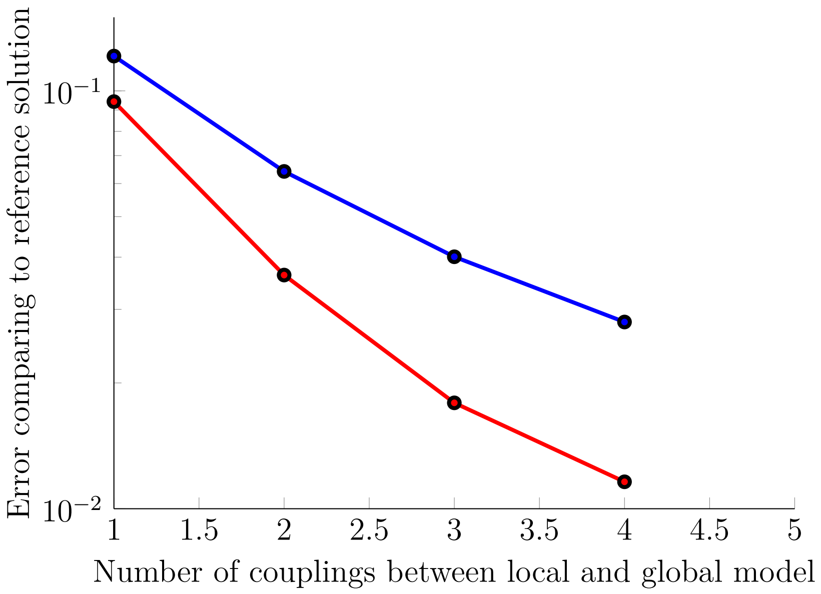


Figure 8.5: Comparison of the errors of the displacements and mixed boundary conditions variations

Global model DOF	4160
Local model DOF	134
Interface DOF	52
Static condensation DOF	4108
Number of iterations	16
Number of couplings	1
Error	$1.9 * 10^{-3}$
Time for calculations	43.47

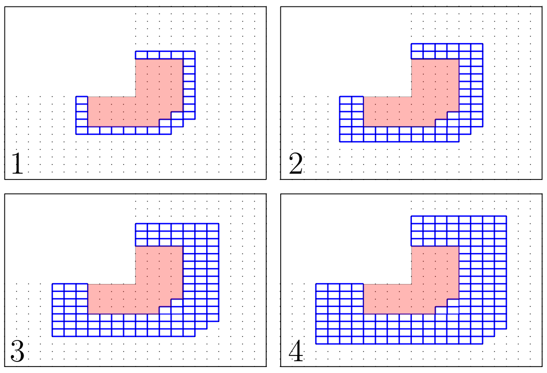


Figure 8.6: Element strips for short scale approximation

1-element strip		2-element strip	
Iterations	Error	Iterations	Error
3	0.1129	5	0.1065
2	0.0581	4	0.0465
2	0.0385	2	0.0252
1	0.0268	1	0.0189

3-element strip		4-element strip	
Iterations	Error	Iterations	Error
5	0.1020	6	0.0977
4	0.0434	5	0.0388
3	0.0229	3	0.0191
1	0.0152	1	0.0129
1	0.0132	1	0.0112

Table 8.3: Number of iterations and error for different element strips at the Schur complement approximation

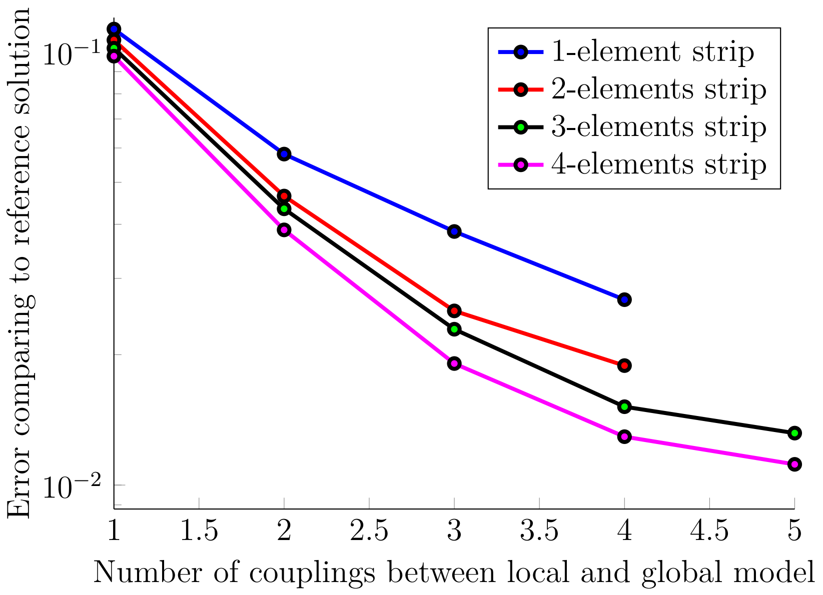


Figure 8.7: Comparison of element strips used for Schur complement two-scale approximation - Error on logarithmic scale

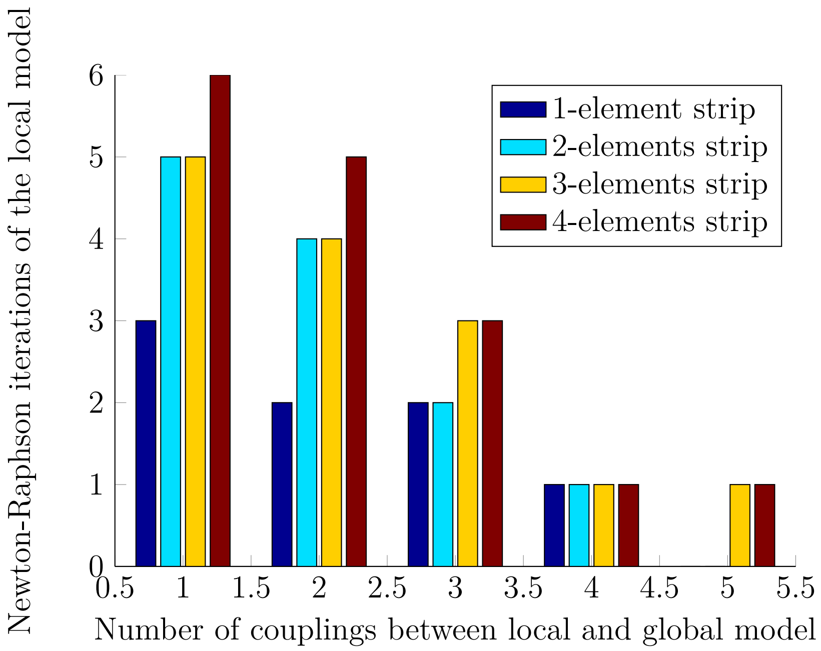
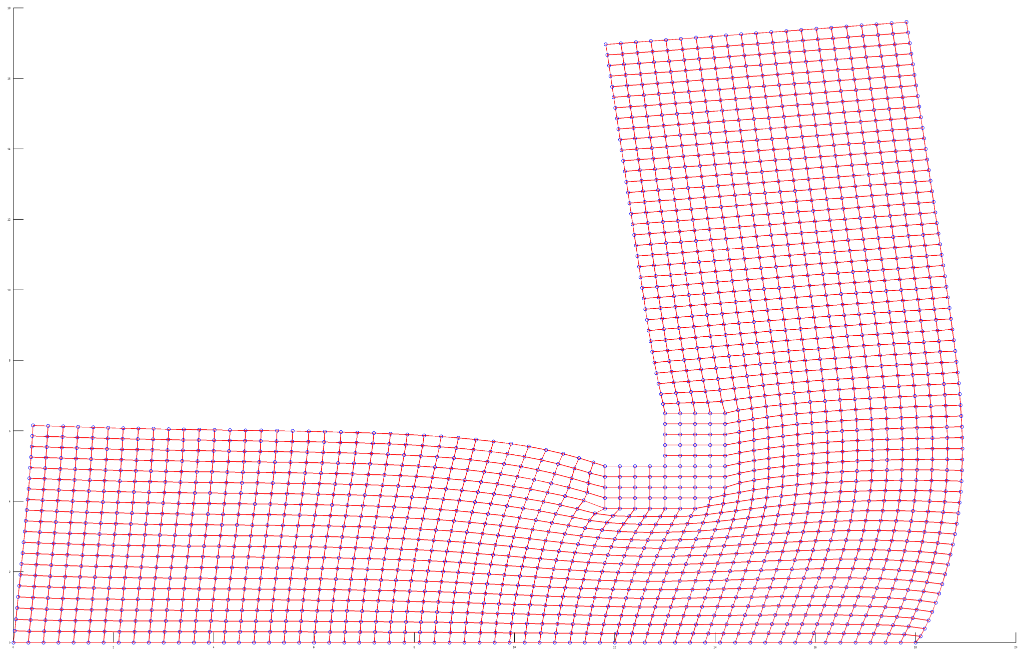


Figure 8.8: Comparison of element strips used for Schur complement two-scale approximation - Number of iterations of the local model

We are doing the same with the rigid body movements, the second parameter in the two-scale approximation of the Schur complement. In Figures 8.9 and 8.10 we can see the rigid body movements as were plotted from the software. Figures 8.11 and 8.12 are the same as those from the short scale approximation but with the rigid body movements as a parameter.


(a) First rigid body movement

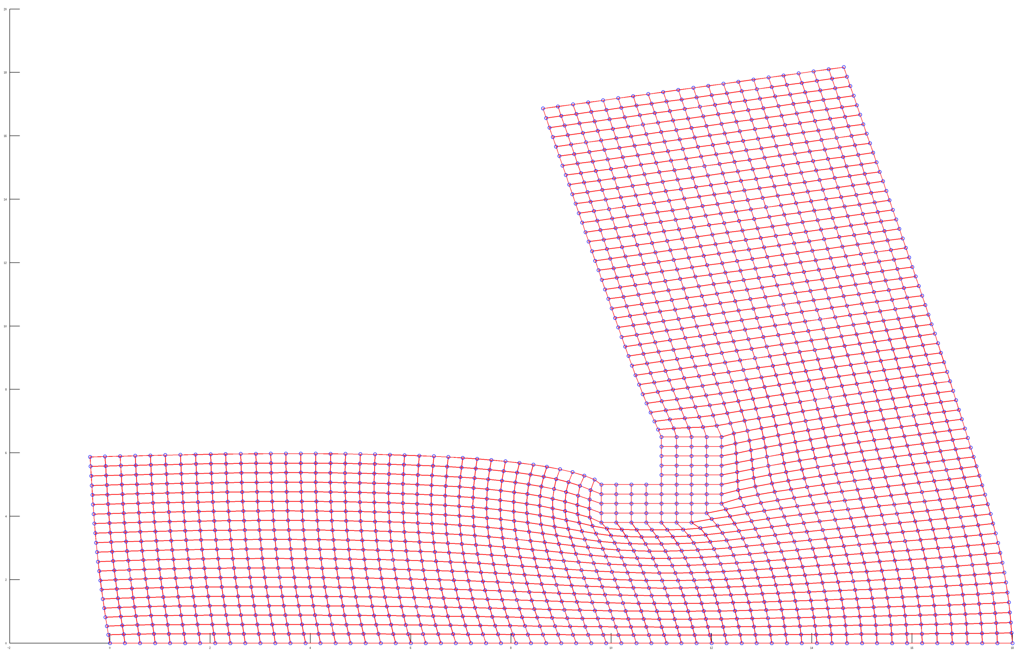
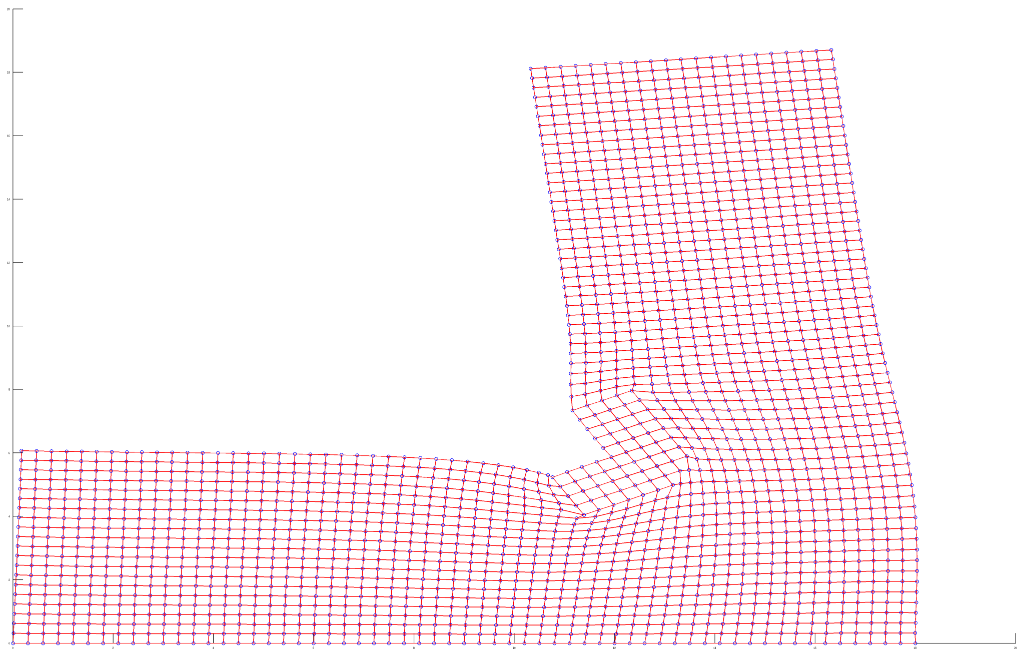

(b) Second rigid body movement

Figure 8.9: Rigid body movements - Translations


(a) Third rigid body movement

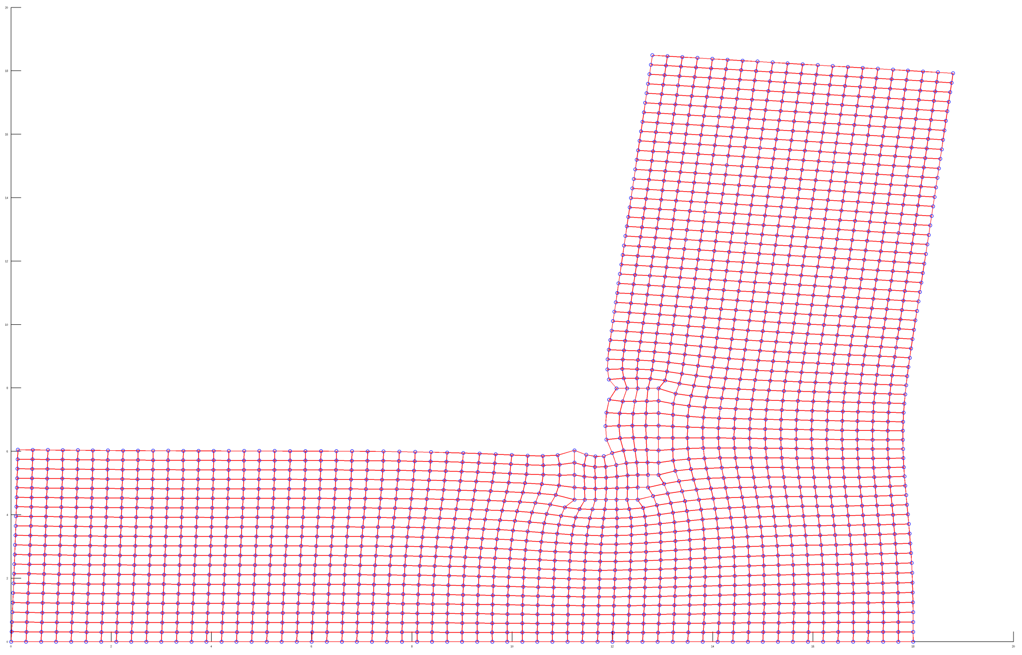

(b) Fourth rigid body movement

Figure 8.10: Rigid body movements - Rotation and distorting mode

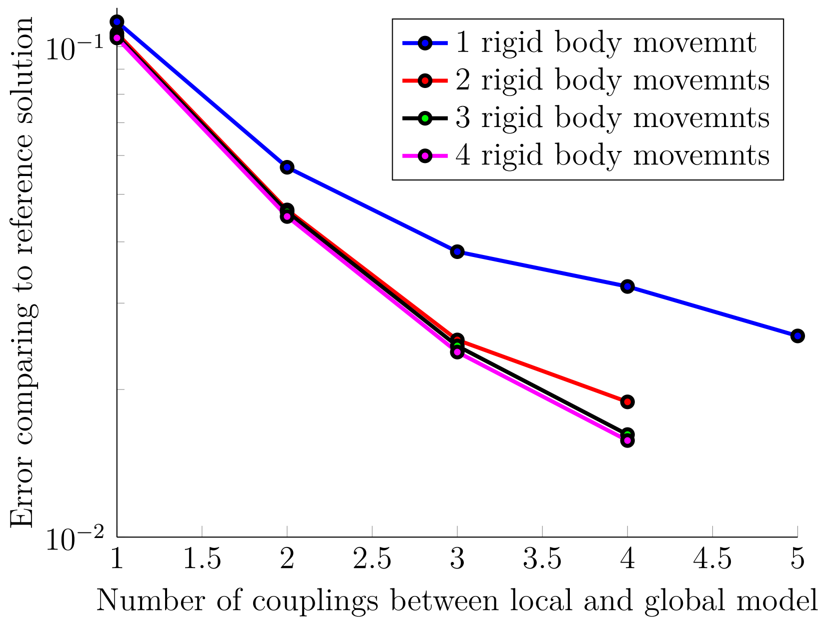


Figure 8.11: Comparison of rigid body movements used for Schur complement two-scale approximation - Error on logarithmic scale

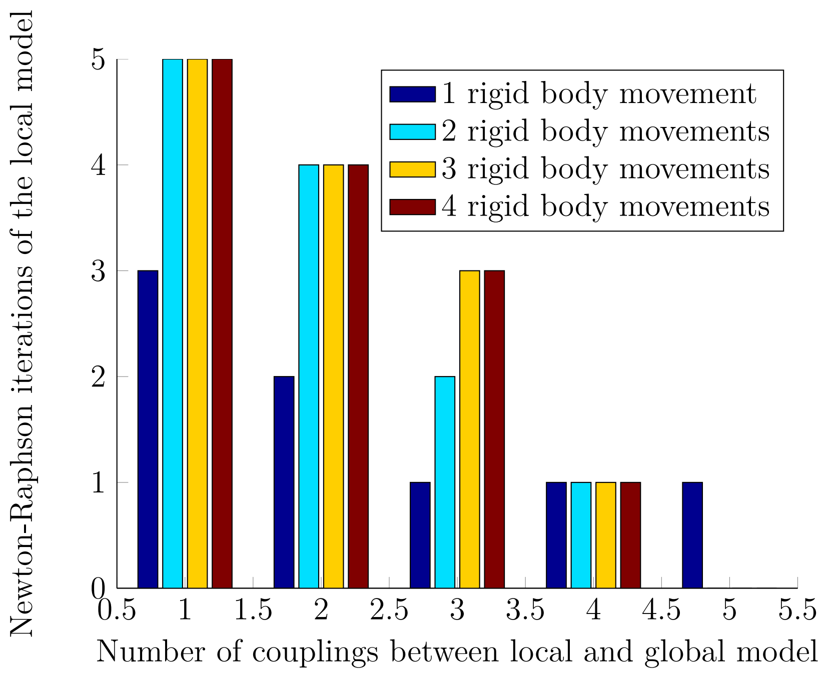


Figure 8.12: Comparison of rigid body movements used for Schur complement two-scale approximation - Number of iterations of the local model

Finally, we perform the analyses with all the combinations (one to four element strips and one to four rigid body movements). In Figure 8.13 the error comparing to the remaining residual is shown. As we can see, the better the approximation is, the least is the error and the remaining residual. Last but not least, Figure 8.14 shows the error compared to the computational time. We can see there that better approximations, take more time to be performed, but the difference of the overall time are a few seconds.

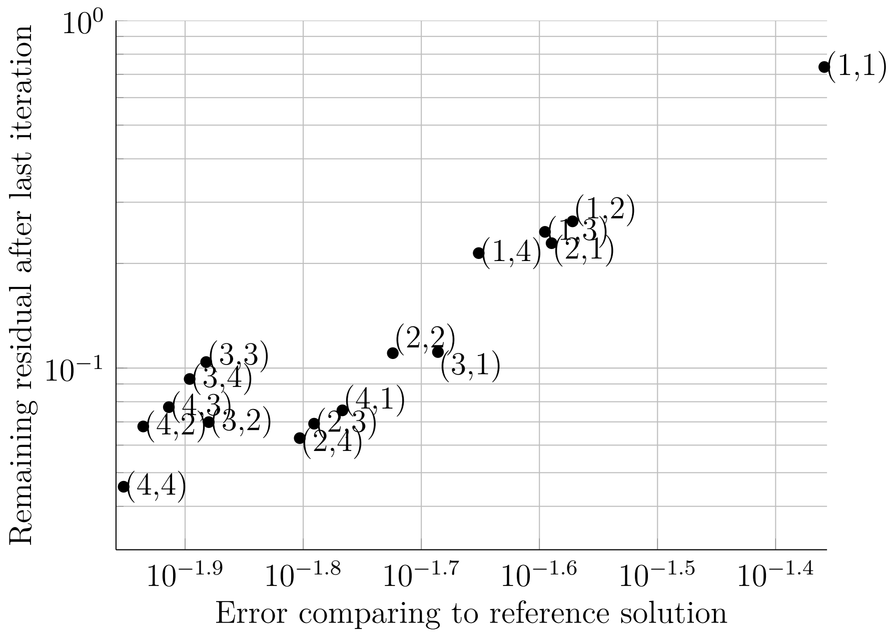


Figure 8.13: Comparison of Schur complement approximations on logarithmic scale

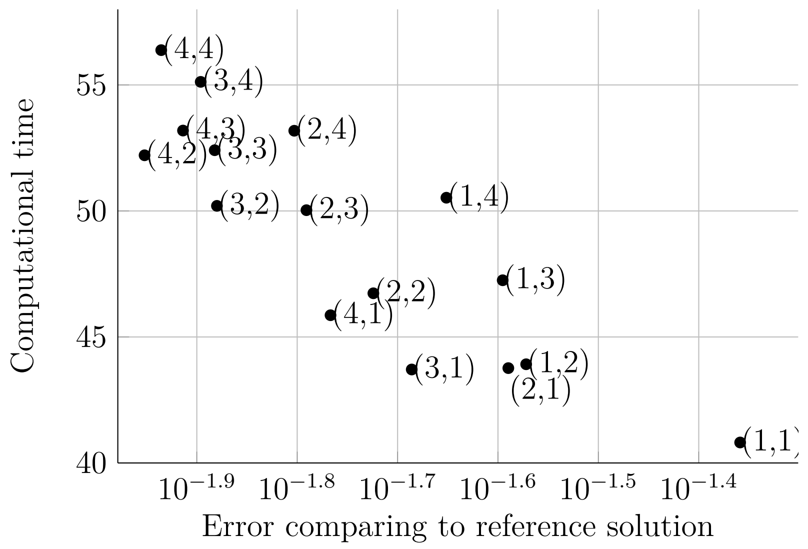


Figure 8.14: Comparison of Schur complement approximations' computational time with error on logarithmic scale

9 Example 2

9.1 Description of the model

In this chapter we are going to study the case of non-conforming interfaces. The model remains the same as in the previous example. It is a common L-shaped model with regular mesh and a linear constitutive law (Figure 8.1). The difference is in the local model that will be created for the two scale method. Materials are also the same as in the previous example. The properties of the model are shown in Table 9.1.

Young Modulus	$210^6 (KN/m^2)$
Poison Ratio	0.3
Yield Stress	$500000 (KN/m^2)$
Finite Element Thickness	0.1
Number of Nodes	2121
Number of Elements	2000
Total Degrees of Freedom	4242
Free Degrees of Freedom	4182

Table 9.1: Properties of the global model in example 2

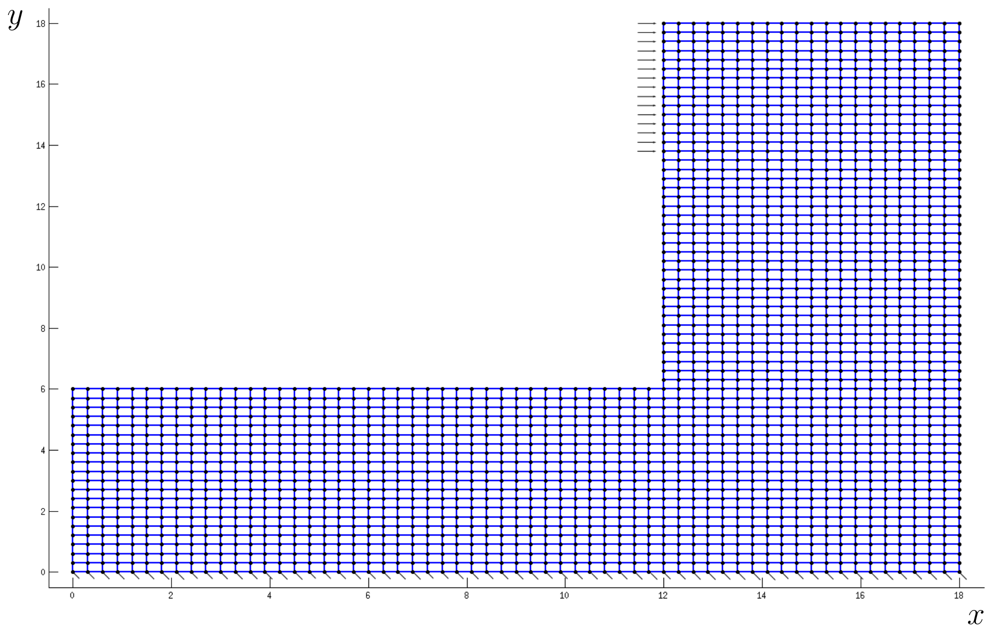


Figure 9.1: The global model in example 2

9.2 Creation of the local model

To study non-conforming interfaces the local model should have different meshing (finer in our case). Consequently, for the purposes of this example the local model has four quadrilateral elements instead of one in the global model. This is a classic example of a multiscale problem were we want to simulate an area of a model in a finer microscale. Once the first linear analysis has been performed in the global (coarse scale) linear model, as in the previous example, the gauss points that have crossed the yield surface are found (we use the von Mises yield criterion). Then the local model is created, with four quadrilateral elements in the place of one as mentioned above. In Figure 9.2 the local model is shown. This model is created in the same area as in example 1 but in a finer scale with more smaller elements.

The connection of the local model with the global model is shown in Figure 9.3. With green color is the global model (coarse scale) while with red color is the local model. The interface is different for the global and the local model. The global interface has the black nodes while the local interface has the black and the red nodes. The global model never changes, this is way the red nodes do not belong to the global model. Also the global model continues to have elements on coarse scale in the local model area (in

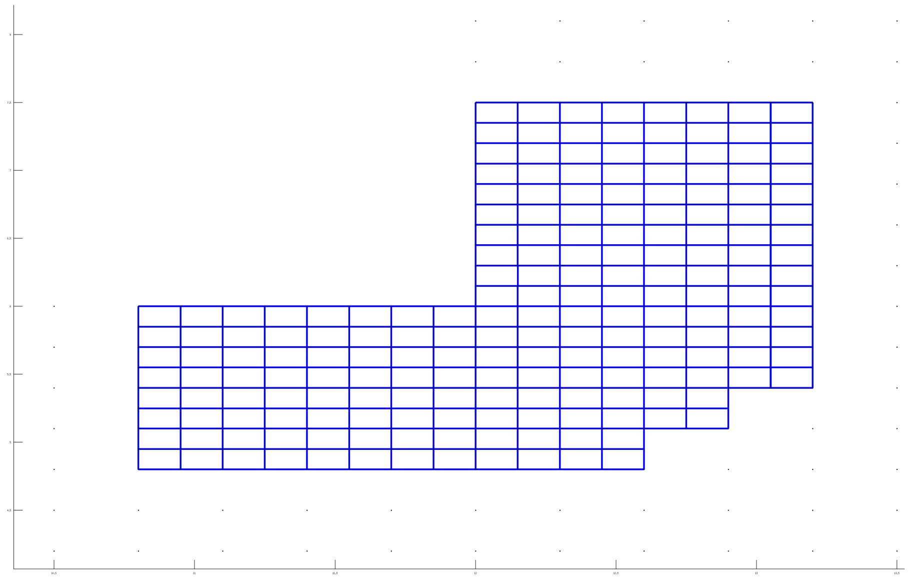


Figure 9.2: The local model

the red area). The hypothetical change of the global model for the creation of such detail in an area would require much effort and difficult interventions especially in 3D problems. This is an advantage of this technique, as the changes are made in a different model and only the interface needs to be defined.

Young Modulus	$210^6 (KN/m^2)$
Poison Ratio	0.3
Yield Stress	$500000 (KN/m^2)$
Finite Element Thickness	0.1
Number of Nodes	213
Number of Elements	196
Total Degrees of Freedom	462
Local Interface Nodes	51
Local Interface Degrees of Freedom	102
Global Interface Nodes	26
Global Interface Degrees of Freedom	52

Table 9.2: Properties of the local model and interface

After the creation of the local model and its interface the matrix $\mathbf{\Lambda}$ is created as described in section 6.2. The non-zero cells are shown in Figure 9.4.

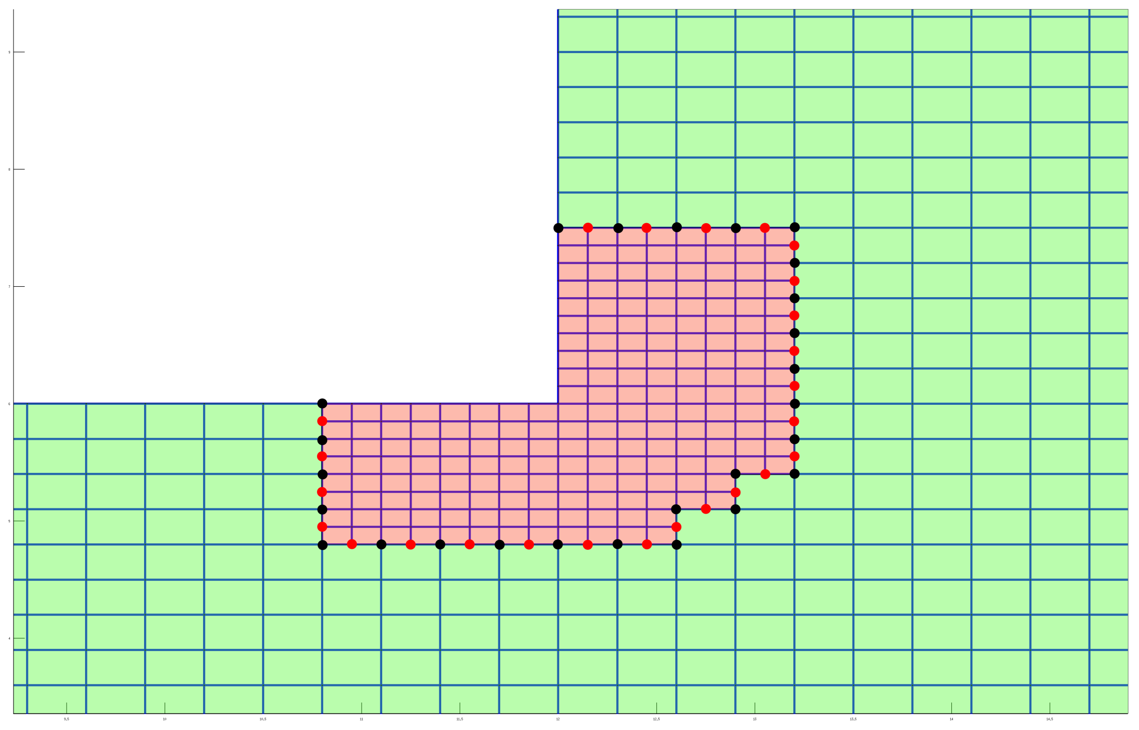


Figure 9.3: The local model, the interface and the global model

The first 52 rows are the common degrees of freedom between the local and the global interfaces so this part of the matrix is an identity matrix. The rest rows are the new degrees of freedom of the local interface. As the new nodes are in the middle of the old ones, these cells have the value 0.5.

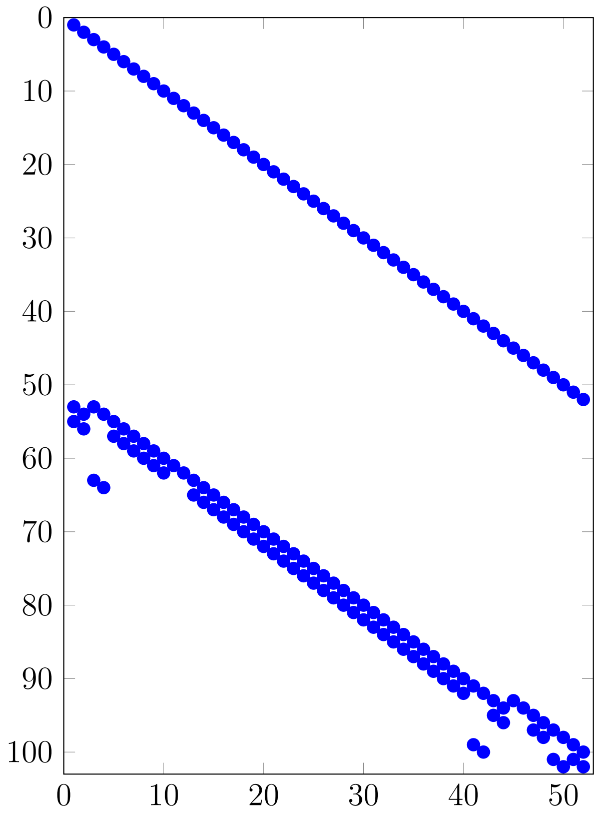


Figure 9.4: Non-zero cells of $\mathbf{\Lambda}$ matrix

9.3 Results

As there is not a reference solution for this problem, we could consider as reference solution, the solution in which the full Schur's complement is calculated explicitly. For this reason the error estimator that is mainly used is the n_u^G. This is because in most situations that the methods would be applied there are no reference solutions already been calculated so the n_u^R error estimator is not available. As there is difference between the local and the global interface the equation (7.3) becomes

$$n_u^G = \frac{\|u_\Gamma^L - \mathbf{\Lambda} u_\Gamma^G\|}{\|\mathbf{\Lambda} u_\Gamma^G\|} \tag{9.1}$$

The residual is also an indicator of the error of the solution. Of course the residual never reaches zero because the iterations are stopped when the normalized forces that are applied to the local model are smaller that the accuracy that has been chosen. If we choose bigger accuracy then the residual and the error estimator are smaller. So these two indicators can give a good perspective of the accuracy of our solution.

First, an analysis with the Schur complement being computed explicitly in an intrusive way was occurred. As expected the coupling was finished after the first local analysis of the model. The results of this analysis are shown in Table 9.3

Global model DOF	4160
Local model DOF	462
Global Interface DOF	52
Local Interface DOF	102
Static condensation DOF	4108
Number of iterations	28
Number of couplings	1
Residual	0.0829
Error n_u^G	0.0467
Time for calculations	57.47

Table 9.3: Analysis results with explicit Schur's complement

Then, different analyses are performed using the two-scale Schur's complement approximation. Table 9.4 shows the results when we change the number of element strips and rigid body movements. The error takes as reference solution the above solution. The two numbers in the 'Type of approximation' rows are the number of element strips and rigid body movements respectively.

We can see that when we create a better approximation to the Schur's complement the solution that we take is much better, but this costs in terms of iterations and time. The decision must be taken according to the computational resources and the demands of the analysis. Furthermore, comparing with the reference analysis and also example 1, the computational cost and time seems to be in same levels. Figures 9.5 and 9.6 provide a visualisation of Table's 9.4 data. In Figure 9.5 the error comparing to the remaining residual is shown. Last but not least, Figure 9.6 shows the error compared to the computational time.

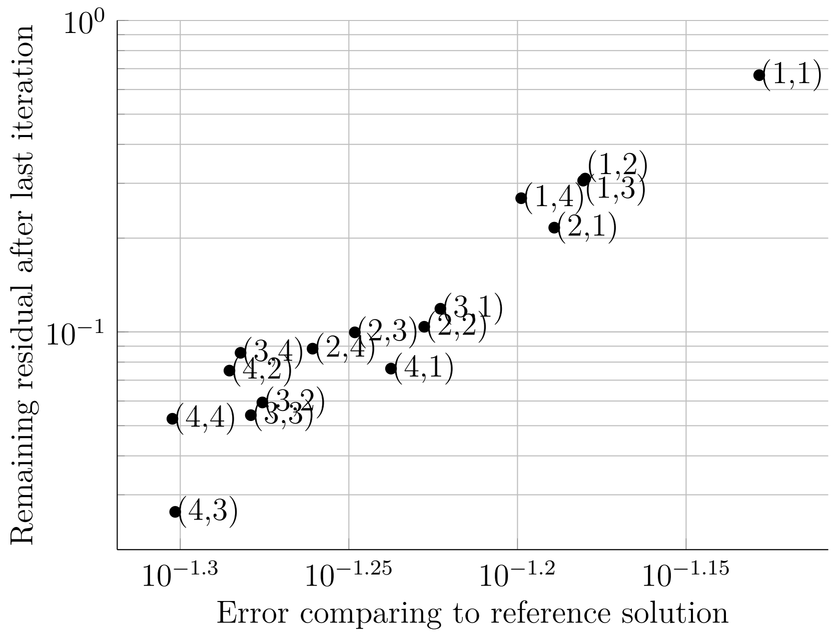


Figure 9.5: Comparison of Schur complement approximations on logarithmic scale

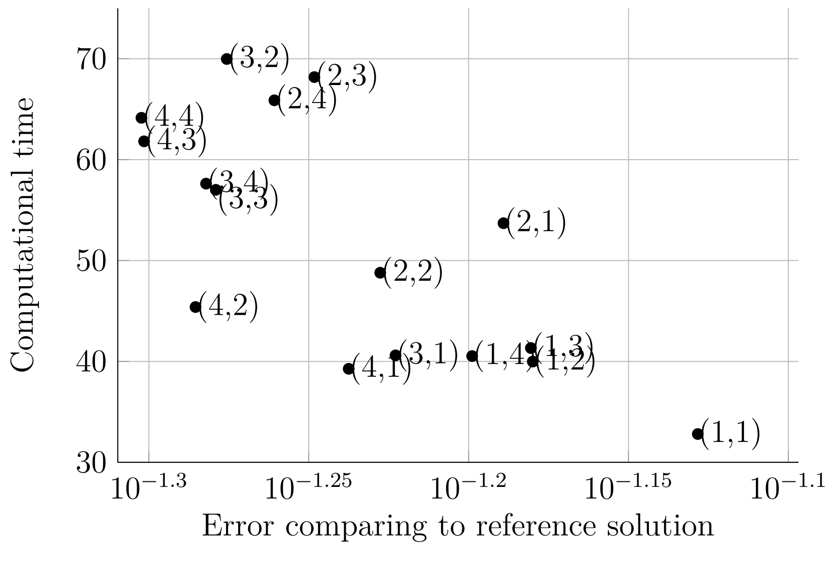


Figure 9.6: Comparison of Schur complement approximations' computational time with error on logarithmic scale

Type of approximation	$(1,1)$	$(1,2)$	$(1,3)$	$(1,4)$
Number of iterations	6	8	8	9
Number of couplings	5	5	5	5
Residual	0.6671	0.3106	0.3059	0.2687
Error n_u^R	0.0744	0.0661	0.0660	0.0632
Time for calculations	32.10	39.29	40.63	39.83

Type of approximation	$(2,1)$	$(2,2)$	$(2,3)$	$(2,4)$
Number of iterations	8	11	13	14
Number of couplings	5	4	5	5
Residual	0.2161	0.1039	0.0997	0.0884
Error n_u^R	0.0647	0.0592	0.0565	0.0548
Time for calculations	52.99	48.09	67.48	65.18

Type of approximation	$(3,1)$	$(3,2)$	$(3,3)$	$(3,4)$
Number of iterations	10	15	16	17
Number of couplings	5	5	5	4
Residual	0.1187	0.0594	0.0540	0.0857
Error n_u^R	0.0599	0.0530	0.0526	0.0522
Time for calculations	39.90	69.27	56.31	56.92

Type of approximation	$(4,1)$	$(4,2)$	$(4,3)$	$(4,4)$
Number of iterations	11	16	18	19
Number of couplings	4	4	5	4
Residual	0.0762	0.0751	0.0264	0.0526
Error n_u^R	0.0579	0.0518	0.0499	0.0498
Time for calculations	38.57	44.69	61.11	63.44

Table 9.4: Analysis results with Schur's complement approximations

10 Conclusions

We can divide the conclusions that derive from this thesis into two categories:

1. Conclusions that refer to the local-global technique. In this section the different types of the Schur complement, the handling of non-conforming meshes and the efficiency of the local-global technique are discussed.

2. Conclusions, reguarding the two-scale Schur complement approximation, that occured through a more detailed study that was carried out on this subject.

10.1 Conclusions for the local-global iteration technique

1. In this thesis a technique was described for the coupling of two models, a global linear and a local non-linear. The major advantage of this technique is that it is a robust and efficient method for problems where a linear domain exhibits confined plasticity (or other non-linear phenomena) in a small critical region. Contrary to other submodeling methods this one does not ignore the global influence of local plasticity [10]. Consequently, it can asses phenomena such as stress redistributions.

2. Through this study it occurred that the explicit computation of the Schur complement seems to provide better results that the two scale approximation in terms of accuracy and speed. However, it is known that in a large problem a static condensation of many degrees of freedom is always avoided because of its computational cost. This is why

the two-scale approximation seems to be a good competitor. Nonetheless the two-scale approximation's errors and time was close enough with the explicit computation's.

3. In addition the mixed boundary variation leads to a convergence of significantly less steps than the displacement variation. This advantage is so huge, that we can conclude that there is no need to use the displacement variation.

4. Finally, a method that handles non-conforming interfaces was introduced successfully. The handling of non-matching meshes between the two models, although conducted in a simple manner, allows the method to address a more general class of locally non-linear problems, and to be a more flexible structural re-analysis and model coupling tool.

10.2 Conclusions for the two-scale approximation

1. In this thesis a deep research was made for the two-scale Schur complement approximation. The combination of the short-scale and the long-scale information is efficient and gives a realistic representation of the stiffness of a large subdomain at realistic costs.

2. The computation of the two-scale Schur's complement approximation is a non-intrusive technique. The ability to apply the techniques in already given models in closed FEM software, in order to introduce locally to them material and geometrical non-linearities and any other details, is a great advantage of this technique. In that way new better analyses can be carried out in older models without the need of remodelling. Furthermore, changes in a model such as a crack, a hole or a reinforcement can be easily introduced to the analysis by creating a new model of the area that changed.

3. The use of this two-scale approximation, as a non-intrusive technique, leads to fast convergence to the iterative coupling strategy introduced in section 4.4. The mixed boundary conditions strategy converges in much faster rate than the displacements variation, allowing the method to be much more efficient.

4. The decision on the number of the element-strips for the short-scale approximation and the number and type of the rigid body movements

for the long scale approximation is a difficult procedure that requires experience. The better the approximation is, the better are the results. However, a better approximation raises the computational cost. The choice depends on the resources, the type of the model and the demands from the analysis.

5. As the two-scale approximation is a representation of the stiffness of a subdomain it can be used as a preconditioner in other methods.

10.3 Problems and future research proposals

Through the study of this thesis we came across severeal problems and ideas for further research. These are suggestions and ideas for the future that there was not sufficient time to be explored and further examined.

1. Although some types of mesh non-conformity where solved, there is need for more research. Cases with totally different interfaces cannot be solved with the proposed technique. As already mentioned, mortar techniques are suitable for these types of problems.

2. For the iterative coupling algorithms there have been already been suggested acceleration techniques, such as the Quasi-Newton and the full tangent Newton algorithms [8]. The Quasi–Newton variant is simple to implement, and significantly improves the convergence rate. The full tangent variant seems even more promising, but requires to process many local and global load vectors at once, and therefore may be too expensive to be used on large models in its current form.

3. The explicit computation of the Schur complement seems to be very promising. However, its high computational cost is a problem. In this case, a connection with domain decomposition methods, such as FETI, can be found. Using such techniques a full condensation could be avoided. The problem would be transformed as a problem of the interfaces of the subdomains and the iteration algorithm would be applied of this interface. In that way, although it would be an intrusive technique, we would avoid the explicit condensation of the global domain.

4. Concerning the coupling strategy, it has currently been tested on problems limited to one single load increment. To handle multiple increments, two possibilities are foreseen: either incrementation is performed

and exchanges are carried out at each increment (like in traditional implicit coupling schemes) or, conversely, exchanges are performed and each global or local problem contains a whole loading history. The latter option, called non-incremental, seems more promising for several reasons. First, it would be more convenient to implement around FEA software which is designed to analyze complete loading histories on a given problem, and not just one specific increment. Second, the non-incremental approach minimizes information transfers between the solvers, and reduces the number of decompositions of the global stiffness matrix, which are time-consuming (during the global correction step, the right-hand side vectors corresponding to the residuals at every time step could be processed at the same time). Finally, it would easily allow using different time steps for the two models.

5. In the current thesis the local model was a model with standard finite elements and material non-linearity. For future work connection with totally different local models is proposed. Local models that handle cracks and crack propagation (i.e. X-FEM) or local models that apply the Element Free Galerkin (EFG) method would be a possible future research topic. This would allow testing advanced models or solution techniques, that are almost impossible to implement in commercial software at the moment, on complex industrial problems that cannot be solved with 'research software' alone.

6. The computations that involves the creation of the two-scale approximation can be possibly programmed in a parallel way resulting in further acceleration. However, the iterations between the two models cannot be expressed as parallel procedures as they require the results of each other in order to proceed.

7. Finally, many more configurations can be thought of, such as multiple areas of interest (i.e. several local models) or multiple zooming levels (i.e. more than two); this could be useful in a number of engineering situations and would give more reliability to the usual 'global-local' simulation scheme, used in many industrial applications of computational mechanics.

Bibliography

[1] O. Allix, P. Kerfriden, and P. Gosselet. On the control of the load increments for a proper description of multiple delamination in a domain decomposition framework. *International Journal for Numerical Methods in Engineering*, 2010.

[2] Olivier Allix. *Approche globale/locale non-intrusive : application aux structures avec plasticité locale.* PhD thesis, École normale supérieure de Cachan, 2009.

[3] C. Bernardi, Y. Maday, and T. Patera. A new nonconforming approach to domain decomposition: the mortar element method. In *Nonlinear Partial Differential Equations and Their Applications, College de France Seminar*, pages 13–51. Pitman, 1991.

[4] P. Cresta, O Allix, C. Rey, and S. Guinard. Nonlinear localization strategies for domain decomposition methods: Application to post-buckling analyses. *Computer Methods in Applied Mechanics and Engineering*, 2007.

[5] Michael Anthony Crisfield. *Non-linear Finite Element Analysis of Solids and Structures*, volume 1. Pineridge Press, 1991.

[6] Y Efendiev and T. Y. Hou. *Multiscale Finite Element Methods: Theory and Applications.* Springer, 2009.

[7] Lionel Gendre, Olivier Allix, and Pierre Gosselet. A two-scale approximation of the schur complement and its use for non-intrusive coupling. *International Journal for Numerical Methods in Engineering*, 87:889–905, 2011.

[8] Lionel Gendre, Olivier Allix, Pierre Gosselet, and François Comte. Non-intrusive and exact global/local techniques for structural problems with local plasticity. *Computational Mechanics*, 44:233–245, 2009.

[9] P. Gupta, J.P. Pereira, D.-J. Kim, C.A. Duarte, and T. Eason. Analysis of three-dimensional fracture mechanics problems: A non-intrusive approach using a generalized finite element method. *Engineering Fracture Mechanics*, 2012.

[10] A. Hund and E. Ramm. Locality constraints within multiscale model for non-linear material behaviour. *International Journal for Numerical Methods in Engineering*, 2007.

[11] Augustus Edward Hough Love. *A Treatise on the Mathematical Theory of Elasticity.* Dover Reprint, 1927.

[12] J. Pebrel, C. Rey, and Gosselet P. A nonlinear dual domain decomposition method: application to structural problems with damage. *International Journal of Multiscale Computational Engineering*, 6:251–262, 2008.

[13] Joris J.C. Remmers Clemens V. Verhoosel René de Borst, Mike A. Crisfield. *Non-linear Finite Element Analysis of Solids and Structures.* John Wiley and Sons Ltd, 2 edition, 2012.

[14] K. Runesson, A. Samuelsson, and L. Bernspang. Numerical techniques in small deformation plasticity. *Numerical Methods in Engineering: Theory and Applications*, 85(1):337–348, 1985.

[15] J. C. Simo and R. J. Taylor. Consistent tangent operators for rate-independent elastoplasticity. *Computer Methods in Applied Mechanics and Engineering*, 48(1):101–118, 1985.

[16] Καραταράκης, Α. Επίλύση φορέων με τη μέθοδο των πεπερασμένων στοιχείων με υποφορείς. Master's thesis, NTUA, 2009.

[17] Παπαγιαννάκης, Α. Επίλυση προβλημάτων πολλαπλών κλιμάκων με τη μέθοδο των υποφορέων. Master's thesis, NTUA, 2013.

[18] Παπαδρακάκης Μ. *Ανάλυση φορέων με τη μέθοδο των πεπερασμένων στοιχείων.* Εκδόσεις Παπασωτηρίου, 2000.

[19] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method for Solid and Structural Mechanics.* Butterworth-Heinemann, 6 edition, 2005.

[20] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals.* Butterworth-Heinemann, 6 edition, 2005.