Aus der Poliklinik für Kieferorthopädie
der Ludwig-Maximilians-Universität München
Direktorin: Prof. Dr. Andrea Wichelhaus

Entwicklung eines Algorithmus zur Simulation von in-vivo-Knochenveränderungen in FE-Programmen

Dissertation
zum Erwerb des Doktorgrades der Zahnmedizin
an der Medizinischen Fakultät
der Ludwig-Maximilians-Universität zu München

vorgelegt von
Mohamad-Sadegh Zoghian

aus
Köln

2017

Aus der Poliklinik für Kieferorthopädie

Klinikum der Ludwig-Maximilians-Universität München

Direktorin: Frau Prof. Dr. Andrea Wichelhaus

Entwicklung eines Algorithmus zur Simulation von in-vivo-Knochenveränderungen in FE-Programmen

Dissertation
zum Erwerb des Doktorgrades der Zahnmedizin
an der Medizinischen Fakultät der
Ludwig-Maximilians-Universität zu München

vorgelegt von

Mohamad-Sadegh Zoghian

aus

Köln

Jahr

2017

Mit Genehmigung der Medizinischen Fakultät
der Universität München

Berichterstatter:	Prof. Dr. Andrea Wichelhaus
Mitberichterstatter:	Prof. Dr. Volkmar Jansson
	Prof. Dr. Hanns Hermann Burgkart
Mitbetreuung durch den promovierten Mitarbeiter:	Dr. hum. biol. Andrew Boryor
Dekan:	Prof. Dr. med. dent. Reinhard Hickel
Tag der mündlichen Prüfung:	16.11.2017

Danksagung

Frau Prof. Dr. Wichelhaus danke ich besonders für die Vergabe des interessanten Promotionsthemas und für die Möglichkeit, diese Arbeit in der biomechanischen Abteilung der Poliklinik für Kieferorthopädie an der LMU München durchzuführen.

Des Weiteren möchte ich Herrn Dr. Boryor für die Betreuung der Arbeit danken. Seine wertvollen Anregungen und Ratschläge habe ich immer geschätzt. Herrn Stapfner danke ich besonders für die Weiterbetreuung meiner Dissertation.

Meinen Eltern danke ich herzlich für ihre fortwährende Unterstützung und ihr Interesse an meiner Arbeit.

Meiner Frau möchte ich von ganzem Herzen für ihre unermüdliche Unterstützung, ihre Liebe und Motivation danken.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungszeichnis

APDL	ANSYS Parametric Design Language
CT	Computertomographie
DVT	Digitale Volumentomographie
FE	Finite Elemente
FEM	Finite Elemente Methode

1. Einleitung

In medizinischen Disziplinen, in denen es um eine möglichst genaue klinische Einschätzung von Knochenumbauten im menschlichen Körper geht, wie zum Beispiel in der Kieferorthopädie, gibt es für den Behandler keine Methode zur exakten Prognose von Veränderungsprozessen lebenden Knochens. Dem Behandler bleibt daher nichts anderes, als sich angesichts dieser Problematik auf seine Erfahrungen zu berufen. Zwar konnten in der Vergangenheit namhafte Persönlichkeiten bezüglich dieser Problematik große Dienste erweisen, indem sie prinzipielle Zusammenhänge feststellten und in Form von Gesetzen beschrieben, wie z.B. das Wolff'sche Gesetz (Wolff 1892). Doch diese Gesetze alleine erlauben nach wie vor keine exakten fallbezogenen klinischen Vorhersagen, sondern nur grobe Tendenzen für die Gestalt und den Verlauf von Veränderungen in lebendem Knochen. Aus diesem Grund kann man sagen, dass manche heute als adäquat anerkannten therapeutischen Maßnahmen möglicherweise nicht vollkommen effektiv sind, weil sie nicht das volle Potenzial knöchernen Wachstums nutzen. Das liegt daran, dass dieses Potenzial momentan nicht genau kalkuliert und vorausgesagt werden kann.

Angesichts dieser Problematik und mit der Erfahrung aus sogenannten FE-Simulationen (Finite Elemente), welche üblicherweise zur optimalen ingenieurmäßigen Neukonstruktion von Gebäuden, Fahrzeugen und im Maschinenbau ihre Verwendung finden, wird versucht, die Veränderungsprozesse lebender knöcherner Strukturen aufgrund der Belastungen, die auf den Knochen wirken, an Computern nachzurechnen und graphisch zu simulieren (Hollister 1994). Diesem Ansatz wurde auch speziell in der Kieferorthopädie zur Simulation von Zahnbewegungen nachgegangen (Meyer 1990). Die Idee, für diese Fragestellung FE-Programme zu nutzen, resultierte daher, dass es eigens für die Simulation von Veränderungen im Knochen noch keine eigene Software gibt und die FE-Methode (FEM) am ehesten als Lösung für dieses Problem in Betracht kommt. Um FE-Programme trotz der Tatsache, dass sie ursprünglich für rein physikalische Fragestellungen konzipiert wurden, nutzen zu können, müssen Algorithmen genutzt werden, mithilfe derer FE-Programme dazu gebracht werden, unter Berücksichtigung der speziell für lebenden Knochen geltenden Bedingungen, möglichst exakte Prognosen für die Veränderung eines vorliegenden Knochens zu berechnen.

Falls dieser Ansatz gelänge, hätte diese neue Technik nicht nur einen praktischen Einfluss auf therapeutische Maßnahmen in der Kieferorthopädie, sondern theoretisch auch auf jede andere

medizinische Disziplin, in der Veränderungsprozesse lebenden Knochens thematisiert sind. Diese Technik brächte zwei wesentliche Neuerungen mit sich:

Zum einen könnte man den, z.B. durch ein Trauma, verloren gegangenen Status quo eines Knochens für einen bestimmten Patienten exakt neu konstruieren, z.B. als Prothesen und Implantate aller Art. Zum anderen hätte die Technik einen weiteren therapeutischen Nutzen; dadurch, dass Veränderungen im Knochen aufgrund von an ihm angebrachten Kräften für die Zukunft vorausberechnet werden können, könnte man bewusst bestimmte Kräfte klinisch so einsetzen, dass sie einen therapeutischen Nutzen hätten, ohne unerwünschte Nebenwirkungen in Kauf nehmen zu müssen. Das betrifft z.B. die Problematik von Wurzelresorptionen bei kieferorthopädischer Behandlung (Wichelhaus 2000).

Die Folgen in Bezug auf einzelne therapeutische Maßnahmen sind in diesem Stadium der Forschung noch nicht abzuschätzen.

Diese Arbeit hatte das Ziel, einen Algorithmus zu entwickeln, mit dessen Hilfe FE-Programme für lebenden Knochen exakte Veränderungen berechnen sollen. Des Weiteren wurde überprüft, wie der entwickelte Algorithmus, in Anbetracht der Entwicklungsstände in zusammenhängenden Fragestellungen, einzuordnen ist.

2. Stand der Technik

2.1 Faktoren der Knochenapposition- und Resorption

Zunächst geht es in diesem Kapitel darum, festzustellen, welche Einflüsse bei Apposition und Resorption von in erster Linie menschlichem Knochen, eine Rolle spielen. Zu dieser Fragestellung finden sich in der Literatur zwei Hauptfaktoren, die im Folgenden näher beschrieben werden. Zum einen wurde festgestellt, dass die Hormone eines Menschen seine Knochenstrukturen beeinflussen. Die Apposition und Resorption von lebendem Knochen ist dabei durch die Aktivität der Osteoklasten und der Osteoblasten determiniert, wobei die Osteoklasten Knochen resorbieren und die Osteoblasten Knochen abbauen. Zum anderen stellte man mechanische Belastungen aller Art als Faktoren für Knochenumbauten im Körper fest. Neben diesen beiden Hauptfaktoren wurden auch weitere Faktoren beschrieben die bei jedem Individuum spezifisch zu beobachten sind.

2.1.1 Hormonelle Einflüsse

Im menschlichen Körper wird die Apposition und Resorption von Knochenmasse durch Hormone reguliert, wie z.B. durch Calcitonin und Parathormon, welche Regulatoren der Aktivität der Osteoklasten und Osteoblasten sind. Die Ausschüttung von Calcitonin führt zu einer Verringerung der Osteoklastenaktivität, was sich makroskopisch als Knochenapposition zeigt. Im Gegenzug wird die Aktivität der Osteoblasten erhöht. Das Parathormon wiederum bewirkt, eine kontinuierliche Gabe vorausgesetzt, als ein Gegenspieler des Calcitonin, eine Steigerung der Osteoklastenaktivität, wodurch vermehrt Knochen resorbiert wird (Hock und Gera 1992).

Bei gesunden Patienten wird die Ausschüttung der Hormone reguliert, indem die Konzentration der Hormone im Blut gemessen wird und bei einer Abweichung vom Normwert, Regulierungsmaßnahmen getroffen werden. Dabei wird den Organen, welche dieses Hormon produzieren – im Fall von Calcitonin und Parathormon der Schilddrüse – über wiederum eigene Hormonkaskaden mitgeteilt, die Produktion entsprechend zu steigern oder zu verringern. Dadurch wird im gesunden menschlichen Körper eine stabile Knochenstruktur gewährleistet.

In pathologischen Fällen kann durch einen deregulierten Hormonhaushalt die Knochenstruktur eines Menschen geschwächt werden, wie z.B. bei Osteoporose bei Frauen nach der Menopause. Osteoporose wird generell als eine multifaktorielle Krankheit deklariert (Jakob et al. 2015), doch die Ursachen der Erkrankung speziell bei Frauen nach der Menopause werden meist durch einen herabsinkenden Östrogen-Spiegel erklärt. Dadurch sinkt auch der Calcitonin-Spiegel im Blut ab, wodurch die Einlagerung von Kalzium in den Knochen immer weniger gewährleistet ist. Als Folge dessen verliert der Knochen an Masse und die trabekulären Strukturen werden weniger dicht. Quantifiziert werden kann dieser Vorgang durch eine Messung der Knochendichte, beispielsweise anhand einer Dual-Röntgen-Absorptiometrie.

Im Gegensatz zur Osteoporose gibt es auch Krankheiten, bei denen durch einen deregulierten Hormonhaushalt überdurchschnittlich viel Knochen aufgebaut wird, wie z.B. die Akromegalie. Eine meist durch einen benignen Tumor im zentralen Nervensystem verursachte Überproduktion an Somatotropin führt dazu, dass vor Verschluss der Epiphysenfugen die Knochen des Menschen überdurchschnittlich lang werden. Nach Verschluss der Epiphysenfugen wirkt sich diese Überproduktion an Somatotropin meist nur noch an den Akren aus.

2.1.2 Mechanische Belastungen

Die Tatsache, dass mechanische Einflüsse die Apposition und Resorption von in-vivo Knochen bewirken, wurde zunächst von Wolff beschrieben (Wolff 1892) und Ende der 1980er Jahre von Frost genauer quantifiziert (Frost 1987). Nachdem Wolff 1892 in seinen Forschungen einen Zusammenhang zwischen der mechanischen Belastung an einem Knochen und der Festigkeit dieses Knochens nachweisen konnte, beschrieb Frost in seinem Proposal, aufbauend auf dieser Beobachtung, vier Intervalle mechanischer Beanspruchung, wobei in jedem Intervall eine bestimmte Reaktion des Knochenstückes festgestellt wurde. Im Bereich von unter 200 µStrain wird aufgrund von zu geringer Belastung Knochenmasse resorbiert. Zwischen 200 µStrain und 1.500 µStrain findet keine Änderung der Knochenmasse statt. Dies ist eine Beobachtung, auf die auch Carter 1982 bereits hingewiesen hatte, die in der Literatur aufgrund der ausbleibenden Veränderung häufig als „lazy zone“ (engl.: träge Zone) bezeichnet wird (Carter 1982). Ab 1.500 µStrain findet erst die Knochenapposition statt. Das

vierte von Frost beschriebene Intervall, in dem keine weitere Erhöhung der Knochenmasse zu erwarten ist und stattdessen der Knochen deformiert werden oder brechen kann, wurde ab einer Belastung von 4.000 µStrain angegeben. Spätere Untersuchungen an Tennisspielern verifizierten Frosts Hypothese über die vermehrte Knochenapposition unter Belastung (Haapasalo et al. 2000). Auch eine Schrumpfung der Knochenmasse wurde in mehreren Studien an Bettlägerigen nachgewiesen (Gross und Rubin 1995; Zerwekh et al. 1998).

Prinzipiell unterscheidet man bei mechanischen Belastungen zwischen Belastungen statischer und dynamischer Art.

Es handelt sich bei statischen Belastungen um die mechanische Beanspruchung von Strukturen – im Falle dieser Arbeit von Knochen – wobei die wirkende Kraft zu nahezu jeder Zeit konstant bleibt. Diese werden bei Menschen in Ruhe vor allem von der eigenen Muskelspannung und der Gravitation der Erde hervorgebracht.

Allgemein wird jedoch angenommen, dass eher dynamische Belastungen Einflüsse auf Knochenumbauten haben, als statische. Kurze Belastungen haben einen größeren Einfluss auf den Knochenumbau, weil Osteozyten gegen regelmäßig auftretende Belastungen zunehmend unempfindlich werden.(Turner und Pavalko 1998)

Von dynamischen Belastungen wird gesprochen, wenn sich die auf den Knochen wirkende Kraft mit der Zeit signifikant ändert. Über die von Frost beobachteten Belastungsintervalle mit den jeweiligen Knochenveränderungen hinaus konnten Robling, Rubin und Skerry beobachten, dass sich dynamische Belastungen durch ihre Frequenz, Änderungsgeschwindigkeit und Dauer auf die Knochenformung auswirken (Robling et al. 2000; Rubin et al. 2004; Skerry 2006).

2.1.3 Andere Einflüsse

Neben der hormonellen Lage und den mechanischen Belastungen spielen unter anderem auch die Ernährung, eine etwaige Medikation des Menschen und auch seine Lebensweise eine Rolle. Eine kalziumarme Ernährung kann beispielsweise dazu führen, dass dem Körper nicht genügend Kalzium zum Einbau in die Knochen zur Verfügung steht und dadurch die Mikroarchitektur der Knochentrabekel degeneriert. Medikamente können bekanntermaßen sowohl eine abbauende, als auch eine aufbauende Wirkung auf Knochenstrukturen haben, wie z.B. die Estradiol-Substitution gegen Osteoporose bei Frauen nach der Menopause. Die

Lebensweise eines Menschen kann insofern Einfluss auf seinen Knochen haben, dass z.B. bei verringerter Vitamin-D-Produktion durch zu wenig Lichtexposition der Haut das vorhandene Kalzium des Körpers nicht in den Knochen eingebaut werden kann.

2.2 Literaturwerte bezüglich der physikalischen Eigenschaften des Knochens

In FE-Programmen lassen sich für Simulationen bestimmte Elementeigenschaften einstellen. In Anbetracht der Fragestellung dieser Arbeit ist es wichtig, die für in-vivo-Knochen festgestellten physikalischen Eigenschaften in Simulationen mit einfließen zu lassen. Diese Werte wurden in der Literatur wie folgt angegeben:

Bezüglich des Elastizitätsmoduls des Knochens stellte Currey fest, dass dieser Wert stark von der Trockenheit und dem Mineralgehalt des Präparates abhängt und deswegen im Vergleich zwischen einzelnen Studien schwankt (Currey 1988).

Der in der Literatur meist angenommene Wert für den Elastizitätsmodul des Knochens wurde von Reilly mit 17.000 N/mm^2 ermittelt (Reilly 1975). In anderen Untersuchungen betrug der Elastizitätsmodul der Kompakta zwischen 7.000 und 16.700 N/mm² (Brear et al. 1990). Der Elastizitätsmodul der einzelnen Trabekel der Spongiosa wurde bei trockenen Präparaten mit 14.130 N/mm² und bei feuchten Präparaten mit 11.400 N/mm² beziffert (Townshend und Rose 1975). Rho wiederum stellte 1997 für den kompakten Knochen einen Wert zwischen 22.500 und 25.800 N/mm² fest, während er für trabekulären Knochen 13.500 N/mm² ermittelte (Rho 1997).

Shahar gibt in seinem Artikel für die Poisson-Zahl des Knochens bei seitlicher Belastung einen Wert von 0,303 und für axiale Belastungen 0,107 an (Shahar 2007).

Als Schubmodul stellte Reilly am menschlichen Femur einen Wert von 3.280 N/mm² fest (Reilly 1975).

Bezüglich der physikalischen Dichte des Knochens stellte man am Femur Werte zwischen 1,73 bis 1,803 g/cm³ fest (Ashman 1988).

2.3 Übertragung der anatomischen Strukturen in das FE-Programm

Wie einleitend beschrieben, hatte diese Arbeit unter anderem das Ziel, zu untersuchen, inwieweit es möglich ist, mit FE-Programmen Knochenmodelle gemäß bekannten Kraftverhältnissen neu zu konstruieren und andererseits auch etwaige zukünftige Veränderungen innerhalb von vorliegenden in-vivo-Knochenmodellen zu berechnen.

Bei ersterer Fragestellung muss untersucht werden, inwieweit FE-Programme dazu in der Lage sind, Knochenmodelle aufgrund der auf sie wirkenden Kräfte neu zu konstruieren. Die Form, von der zur Neukonstruktion eines Modells ausgegangen werden muss, sollte dabei eine sein, auf der einerseits alle wirkenden Kräfte an den physiologischen Stellen angebracht werden können. Andererseits sollte die Form jedoch möglichst einfach gehalten werden, um prüfen zu können, ob die jeweiligen Programme aus einer neutralen ergebnisoffenen Form in eigenständiger Weise die gewünschte physiologische Form konstruieren können.

Bei der zweiten Fragestellung ist es jedoch unerlässlich, das real vorliegende Knochenstück in ein FE-Programm zu übertragen, um anhand dieser dann Simulationen durchführen zu können. Es geht zum einen um die Übertragung des Ausgangsmodells des Knochens, wofür schon weitgehend überzeugende Lösungen gefunden wurden.

Zum anderen geht es, sowohl in dieser Fragestellung, als auch in der Fragestellung von Neukonstruktionen, um die Übertragung der an einem Knochen anliegenden Kräfte, also der Randbedingungen, in das FE-Programm, was sich bisher als schwieriger erwies.

Prinzipiell muss jedoch an dieser Stelle auch gesagt werden, dass Faktoren jeglicher Art, die einen Einfluss auf Umbauprozesse im Knochen haben, letztendlich nur als mechanische Einflüsse auf das Knochenstück, in Form von physikalisch anzugebenden Größen – Kräfte [N] oder Drehmomente [Nm] –, in der FE-Rechnung berücksichtigt werden können. Etwaige andere nicht mechanische interpretierbare Einflüsse, wie z.B. hormoneller Art, können nach dem letzten Stand der Softwaretechnik in einem FE-Programm nicht mit eingebracht werden.

2.3.1 Übertragung der Knochenform in das FE-Programm

In FE-Programmen lassen sich in den jeweiligen Benutzeroberflächen prinzipiell zwei- und dreidimensionale Formen sowohl mit vordefinierter Geometrie, also z.B. Kreise, Kugeln, Prismen, Quadrate oder Kuben, als auch mit einer vom Benutzer frei zu gestaltenden

Geometrie virtuell und optisch sichtbar konstruieren. Dabei können Formen auch miteinander kombiniert werden, um komplexere Modelle zu erstellen. Um Modelle, die menschlichen Knochen ähneln zu konstruieren, ist die Methode, in der der Benutzer die Form frei gestalten muss, nicht genau genug, weil sie die Form des Knochens niemals exakt wiedergibt. Die Methode mit der Kombination vordefinierter Formen ist zu zeitaufwendig und entspricht letztendlich trotzdem nicht dem Status quo des echten Knochens. In derlei Fragestellungen bietet es sich also eher an, zu versuchen, die zu untersuchenden Knochenmodelle in das jeweilige FE-Programm einzulesen. Dahingehend gibt es die Möglichkeit, CT- (Computertomographie), Mikro-CT- und DVT-Datensätze (Digitale Volumentomographie) des Knochens, die als digitale Ausgangsdaten dienen können, mittels sogenannter Segmentierungsprogramme in fertige FE-Modelle umzuwandeln, mit denen in FE-Programmen weiter verfahren werden kann. Auf die Frage, was ein FE-Modell genau ist, soll in Kapitel 2.3.1.2 genauer eingegangen werden. Für die folgenden Erläuterungen reicht es zunächst aus, anzunehmen, ein FE-Modell sei ein Modell vom Knochen, das mit Hilfe eines Rechners graphisch dargestellt werden kann, und auf das man Kräfte anbringen und entsprechende Veränderungen simulieren kann.

2.3.1.1 Segmentierung und die Verwendung von Segmentierungs-programmen

Per Definition bezeichnet Segmentierung „die Erzeugung von inhaltlich zusammenhängenden Regionen durch Zusammenfassung benachbarter Pixel oder Voxel entsprechend einem bestimmten Homogenitätskriterium“. (Wikipedia-Autor) (siehe Abbildung 1).

Abbildung 1: Avizo (Segmentierungssoftware), Firma FEI Visualization Sciences Group, Burlington, USA – Man sieht, dass die Software aus einem CT-Datensatz unter anderem die Umrisse der Molaren und des Nasenseptums erkannt hat.

Die Notwendigkeit zur Nutzung von Segmentierungsprogrammen ist nicht nur in der softwarebedingten Transferierung von radiologischen Daten hin zu den FE-Programmen begründet, sondern auch das „Vernetzen“ (siehe Kapitel 3.1.1) des Modells ist ein ebenso notwendiger Arbeitsschritt, den Segmentierungsprogramme leisten können. Wenn man davon ausgeht, dass ein radiologischer Scan von einem Knochen vorliegt, sei es z.B. ein DVT eines Unterkiefers, so ist auf diesem DVT nicht nur der Unterkieferknochen zu sehen, sondern zwangsläufig immer auch alle Strukturen, die den Unterkiefer zum Zeitpunkt des Scans umgaben. Zum Einlesen des Unterkiefers in das FE-Programm wird jedoch nur der Unterkiefer benötigt, ohne die ihn umgebenden Strukturen, wie z.B. Muskeln, Sehnen, Faszien, Gefäße, Nerven oder auch Artefakte, denn die erwähnten mit eingescannten Strukturen sind für die Erstellung eines reinen Knochenmodells primär störend und sollen deshalb auch nicht Teil des FE-Modells werden.

Prinzipiell soll die Kalkulation der Veränderungen eines Knochens im FE-Programm so ablaufen, dass von einem reinen Knochenmodell in Form eines FE-Modells ausgegangen wird, und anschließend auf diesem Modell physikalische Randbedingungen wie Kräfte und

Lager, die in vivo Muskeln und Gelenken entsprechen, angebracht werden. Aus diesem Grund müssen zunächst mithilfe eben dieser Segmentierungsprogramme die einzelnen Strukturen, zumindest jedoch der Knochen vom restlichen Gewebe, unterschieden werden, um anschließend für die Erstellung des Knochenmodells die oben genannten Strukturen entfernen zu können, sodass letztendlich nur noch der Knochen übrig bleibt. Je höher die Auflösung der radiologischen Daten, desto genauer kann man gewünschte Bereiche segmentieren. Abbildung 2 zeigt den fertig segmentierten Scan eines menschlichen Schädels ohne die erwähnten störenden Einflüsse.

Zu den gängigen in der Biomechanik verwendeten Segmentierungsprogrammen, die auch in der Kieferorthopädie verwendet werden, zählen die Programme Amira[1], Materialise[2] und Simpleware[3].

Technisch gesehen funktioniert die Segmentierung in den gängigen Segmentierungsprogrammen auf zwei Arten, die automatische Segmentierung und die manuelle Segmentierung.

2.3.1.1.1 Automatische Segmentierung

In der automatischen Segmentierung werden die unterschiedlichen Graustufen in der Darstellung der einzelnen Strukturen genutzt und gleiche oder sehr ähnliche Graustufen automatisch jeweils als ein bestimmter Strukturtyp interpretiert. Am Beispiel des Unterkiefer-DVTs hieße das z.B., dass Schmelzareale von Zähnen aufgrund ihrer höheren Radioopazität andere eher verschattete Graustufen haben (siehe z.B. Abbildung 1), als der Knochen, welcher eher aufgehellt erscheint. Auf dieser Grundlage interpretiert die Software eigenständig z.B. alle Schmelzareale der Zähne als einen Gewebetyp und unterscheidet diese vom Gewebetyp des Knochens. Analog gilt dieses Prinzip auch für alle anderen im DVT abgebildeten Anteile, wie z.B. für die Kompakta, Dentin, Nerven und Gingiva, je nachdem wie technisch akkurat das Segmentierungsprogramm arbeitet.

---

[1] Firma FEI Visualization Sciences Group, Burlington, USA

[2] gleichnamige Firma, Leuven, Belgien

[3] gleichnamige Firma, Exeter, UK

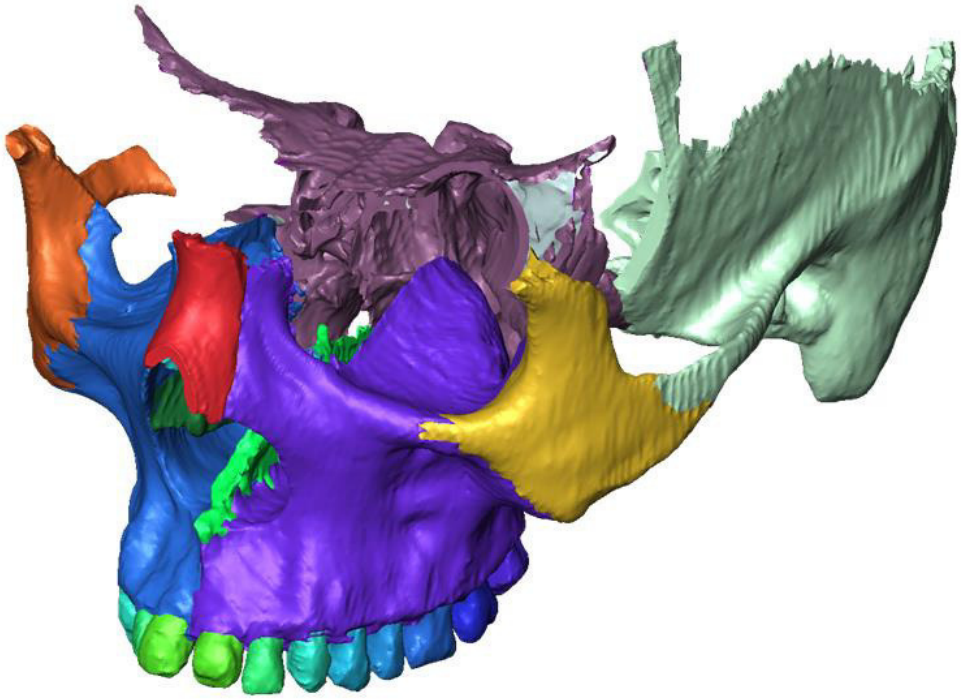


Abbildung 2: Amira – Zu erkennen sind fertig segmentierten Knochenstücke und Zähne eines menschlichen Schädels. Die Nutzung der verschiedenen Farben für die unterschiedlichen Segmente impliziert die Tatsache, dass diese Segmente im Programm technisch voneinander unterschieden werden können.

Die Anwendung der automatischen Segmentierung ist jedoch nicht immer sinnvoll, denn manchmal können Segmentierungsprogramme, weil bestimmte Graustufen sich zu ähnlich sind, oder wenn Artefakte vorliegen, unterschiedliche Gewebetypen nicht voneinander unterscheiden. Dies ist z.B. dann der Fall, wenn man zum Beispiel einen DVT-Datensatz von einem Kiefer hat, in dem zum Zeitpunkt des Scans ein Aligner eingegliedert war. Dort kann es sein, dass die automatische Segmentierung den Aligner, aufgrund der ähnlichen Graustufe zum Knochen, vom Knochen nicht unterscheiden kann und deswegen nicht zufriedenstellend segmentiert. In einem solchen Fall bleibt nach dem letzten Stand der Segmentierungstechnik einzig die Option, den radiologischen Scan in der Segmentierungssoftware manuell zu segmentieren (Vallaeys 2015).

2.3.1.1.2 Manuelle Segmentierung

Das manuelle Segmentieren nimmt, je nach Größe und Auflösung des zu untersuchenden Modells unverhältnismäßig mehr Zeit in Anspruch, als die automatische Segmentierung. Sie ist aber in der Genauigkeit der Segmentierung der automatischen Technik überlegen, da der

Nutzer hierbei die vollkommene Kontrolle über das Segmentierungsgeschehen hat. Die Arbeit, die bei der automatischen Segmentierung aufgrund der Ähnlichkeit oder der Verschiedenheit der Graustufen der einzelnen Areale durch eine Software übernommen wird, hat bei der manuellen Segmentierung der Nutzer selber zu bewältigen, in dem er Areal für Areal den gesamten radiologischen Scan durchscrollt und selber definiert, welche Areale zusammengehören und welche sich unterscheiden.

2.3.1.2 Erstellung eines FE-Modells

Nach erfolgter Segmentierung werden, unabhängig davon, ob automatisch oder manuell segmentiert wurde, alle Areale, die nicht zum klinischen Modell dazugehören sollen, entfernt. Der nächste Schritt ist dann die Umwandlung des übriggebliebenen radiologischen Scans, in ein FE-Modell. Dieser Schritt ist unverzichtbar, um mit FE-Programmen rechnen zu können und wird auch „Vernetzen“ genannt.

Die Erstellung des FE-Modells kann prinzipiell sowohl im Segmentierungsprogramm, als auch im FE-Programm, als auch in einem dritten eigens zu diesem Zweck genutzten Vernetzungsprogramm erfolgen. Sofern das Netz des FE-Modells nicht im FE-Programm erfolgt, muss anschließend, um an diesem Modell Berechnungen durchführen zu können, dieses in das FE-Programm transferiert werden.

Das Netz der zusammenhangenden Elemente wird vom jeweils verwendeten Programm automatisch generiert. Wie beispielsweise in Abbildung 3 zu sehen ist, wurden spongiöse Anteile eines Unterkiefers mit feineren Elementen vernetzt, als die Kortikalis, welche mit größeren Elementen vernetzt wurde. Die genaue Definition des Begriffs „Element“ erfolgt in Kapitel 3.1.1.1.

Da die Finiten Elemente auch die Materialeigenschaften der zu berechnenden Geometrie darstellen sollen, müssen der Elementtyp und das Material innerhalb der Eigenschaften der verwendeten Elemente eingestellt werden (siehe Kapitel 3.1.1.1.1).

Die Genauigkeit der Simulationsergebnisse der FEM ist von der Netzdichte abhängig. Regionen mit hohen Spannungskonzentrationen sollten ein feines Netz besitzen (siehe Abbildung 4).

Alle gängigen Vernetzungsprogramme können unter der angegebenen Quelle nach gelesen werden (Schneiders).

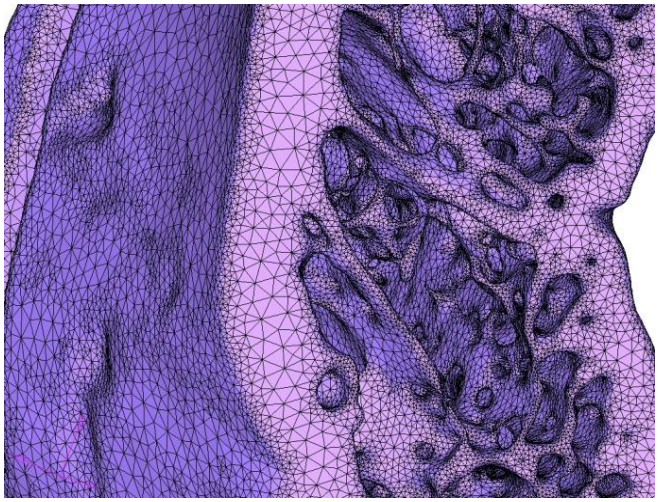


Abbildung 3: Amira – FE-Modell eines Teiles eines Unterkiefers. Zu sehen sind unterschiedlich große Elemente, je nachdem wie fein die zu vernetzende Struktur ist

2.3.2 Übertragung der Randbedingungen in das FE-Programm

Anders als bei der Erstellung und Übertragung des Knochenmodells in das FE-Programm, zeichnet sich die Übertragung der klinischen, am Knochen vorliegenden, Randbedingungen in das FE-Programm als schwierig ab. Im Wesentlichen geht es bei den Randbedingungen um zwei Komponenten: Kräfte und Lager. Die Kombination aus diesen beiden Komponenten ergibt Drehmomente, die ebenfalls eine wichtige Komponente bei der Kalkulation darstellen. Die Einschätzung der Position der Lagerstellen gestaltet sich meist einfach. Lagerstellen entsprechen klinisch den Gelenken und sollten im FE-Modell jeweils an dem zentralen Punkt gesetzt werden, um das sich der Gelenkkopf dreht.

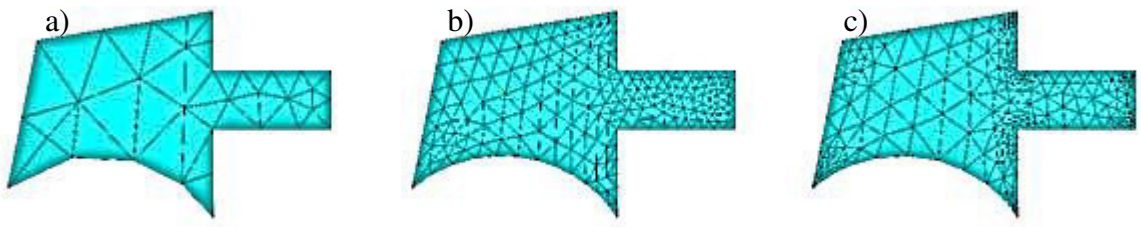


Abbildung 4: ANSYS – Vernetzungsarten a) grobe Vernetzung b) feine Vernetzung c) verfeinertes Netz nur an den Stellen, wo es notwendig ist

Die wirkliche Schwierigkeit bei der Übertragung der klinischen Situation in das FE-Programm stellt also die exakte Quantifizierung der Kräfte dar. Selbst ein vermeintlich einfacher Kaumuskel wie der Musculus masseter kann nicht durch einen einzigen Vektor, der von der Tuberositas masseterica zum mittleren Drittel des Arcus zygomaticus führt, dargestellt werden. Diese Annahme stimmt mit der Klinik nicht genau genug überein und würde das Ergebnis deshalb verfälschen. Der besagte Muskel besteht nämlich aus sehr vielen Muskelfasern, von denen manche weiter profund verlaufen und andere weiter superficial. Eine Dissertation, für die die Dicke des Musculus Masseter an insgesamt 29 erwachsenen Probanden gemessen wurde, kam beispielsweise zu dem Ergebnis, dass dieser Muskel im Durchschnitt im entspannten Zustand 10,0 mm dick ist (Regber 2002). Diese Tatsache führt dazu, dass es, zur Darstellung der Kräfte dieses Muskels, sehr viele Vektoren mit teilweise voneinander abweichenden Richtungen geben muss, von denen manche von weiter profund entspringen als andere. Das gleiche Prinzip beobachtet man bei diesem Muskel auch in dorso-ventraler Richtung.

Der Ansatz, alle Muskelfasern eines Muskels durch Vektoraddition durch einen einzigen Vektor zu ersetzen, führt nicht zum Ziel. Das liegt daran, dass zum einen Ansatz und Ursprung von Muskeln nicht punktförmig, sondern eher flächig sind und zum anderen, bestimmte Anteile von Muskeln durch den Menschen bei jeder Bewegung anders gesteuert werden können. Es ist immer der Mensch, der bewusst oder unbewusst entscheidet, wann und wie stark er bestimmte Muskelanteile in einer bestimmten Bewegung nutzt und das ist in zwei nacheinander folgenden Bewegungen genau genommen nie gleich.

Bei der Simulation von Zahnbewegungen im Kiefer wird diese Aufgabe noch dadurch erschwert, dass das Programm hier zwischen initialer und langfristiger kieferorthopädischer Bewegung unterscheiden muss. Während die initialen Zahnbewegung durch das Streckungsverhalten des parodontalen Ligamentes begründet ist, handelt es sich bei der langfristigen orthopädischen Zahnbewegung um einen biologischen Prozess, bei dem Knochen-Apposition und –Resorption stattfinden (Kojima 2014).

Es ist ebenfalls zu beachten, dass für die Erstellung eines Knochenmodells, aufgrund der Kräfte, die auf diesen Knochen wirken, nicht nur die Beträge und Richtungen aller Kräfte zu kennen, die in einem Augenblick auf diesen Knochen einwirken verantwortlich sind. Wie in Kapitel 2.1.2 bereits erwähnt, gilt es als erwiesen, dass für die Form des Knochens nicht nur die Kräfte in einem einzigen Augenblick zuständig sind, sondern dass die Kräfte im gesamten

zeitlichen Verlauf einer Bewegung, also als dynamische Belastungen, für die Form eines Knochens zuständig sind. Diese Annahme bedeutet, dass zur exakten Simulation eines Modells alle Richtungen und alle Beträge aller Kräfte zu allen Zeiten einer Bewegung und das für jede Art von Bewegung, oder besser gesagt, für jeden Lastschritt kennen müsste. Auf den Begriff Lastschritt wird in Kapitel 4.1.4 näher eingegangen.

Die benannten Schwierigkeiten führen letztendlich dazu, dass auch kleinste Fehler in der Beobachtung der Muskelkräfte in der Simulation Fehler aufkommen lassen, die die Endresultate verfälschen (Boryor und Sander 2008). Diese Problematik hat in der Geschichte der Biomechanik schon immer eine wichtige Rolle gespielt, was auch dazu führte, dass bisher viele Ansätze unternommen wurden, Muskelkräfte exakt quantifizierbar zu machen (Chan 2015). Im Folgenden soll deshalb ein kurzer Überblick über die verschiedenen Ansätze von in vivo Kräftemessungen, jeweils im Hinblick auf das Messverfahren und die Qualität der Messergebnisse gegeben werden.

2.3.2.1 Elektromyografie (EMG)

Die Elektromyografie ist ein Verfahren, mit dem die neuronale Aktivität, in Form von elektrischen Potenzialen, von ganzen Muskeln und auch von einzelnen Muskelfasern gemessen werden kann. Für die Messung gibt es zwei Methoden: Die Messung über Hautelektroden und die Messung mittels Nadelelektroden. Bei der ersten Methode werden Elektroden, die über Kabel oder auch kabellos mit einem Messgerät verbunden sind, an die Hautoberfläche in der Region des zu messenden Muskels angebracht. Dort wird bei Muskelaktivität, aufgrund der Aktionspotenziale innerhalb der Muskelzellen eine elektrische Schwankung ermittelt, die auch grafisch dargestellt werden kann. Bei der Messung mittels Nadelelektroden hingegen werden Nadeln verwendet, die je nach Hersteller unterschiedliche Durchmesser haben. Mithilfe dieser Nadeln kann, dadurch dass man sie transdermal in die zu messende Muskelfaser einführt, auch das Aktionspotenzial einer einzelnen Muskelfaser gemessen werden (Fuentes 2016).

Unabhängig von der gewählten Messmethode gibt die Messung einen Aufschluss über die potenzielle Aktivität eines Muskels. Diese Information erlaubt jedoch nicht unbedingt einen exakten Rückschluss auf den Betrag seiner Kraft, denn letztendlich wird nur die elektrische

Aktivität an der motorischen Endplatte gemessen. Die Richtung der Kraft kann dabei indirekt aufgrund der Lage der messenden Elektroden ermittelt werden.

2.3.2.2 Anatomische Kräftemessung mithilfe der Magnetresonanztomographie

In magnetresonanztomographischer Bildgebung können Weichteile, wie z.B. Muskeln, sehr genau und dreidimensional dargestellt werden. Außerdem fällt bei dieser Art der Bildgebung keine belastende Röntgenstrahlung an. Deswegen besteht die Möglichkeit, mithilfe dieser Technik zu versuchen, die Kraftverhältnisse innerhalb der abgebildeten Muskeln aufgrund des Muskelquerschnittes zu quantifizieren. Prinzipiell stimmt zwar die Annahme, dass die Anatomie von Muskeln bei der Magnetresonanztomographie gut befunden werden kann. Forschungsergebnisse haben auch zeigen können, dass der Querschnitt eines Muskels über eine bestimmte Formel in Relation zu der Kraft steht, die dieser potenziell aufbringen kann (Hackenberg 2005). Jedoch ist die aufbringbare Kraft auch abhängig vom jeweiligen Hebelarm, der wiederum bei jedem Individuum unterschiedlich ist (Hackenberg 2005).

2.3.2.3 In-vivo-Kräftemessung

In der Literatur finden sich auch Ansätze, in denen versucht wurde, am lebenden Versuchsobjekt Belastungen direkt am Knochen zu messen. Bergmann z.B. beschrieb 1997 In-vivo-Kräftemessungen am Hüftgelenk. Dabei nutzte er die Vier-Kanal-Telemetrie zur Übertragung von Daten, die von einer im lebenden Menschen hineinoperierten Messprothese gemessen wurden, auf eine extrakorporal liegende Apparatur. Die Messungen an sich fanden mithilfe sogenannter Dehnungsmessstreifen statt, die in der Prothese eingebaut waren. Dadurch, dass je Prothese drei dieser Dehnungsmeßstreifen verwendet wurden, konnten mithilfe der Matrix-Methode die Kräfte, die bei bestimmten Aktivitäten auf die Prothese wirkten, insgesamt je zu einem Vektor zusammengerechnet werden. Außerdem konnten aufgrund der gemessenen Kräfte auch die jeweiligen Drehmomente berechnet werden. Gemessen wurden hauptsächlich die Maximalkräfte auf den Prothesenkopf bei Standard-

Aktivitäten wie z.B. Gehen, Stehen, Sitzen, Treppensteigen, das Gehen mit Stockstützen, Krankengymnastik[4], Joggen und weitere Sportarten (Bergmann 1997e).

Die Meßgenauigkeit betrug bei den damaligen Versuchen, so Bergmann – etwaige Fehler bei der Signalübertragung mit eingerechnet – maximal 1,5% des jeweiligen Messbereiches bei Kraftkomponenten zwischen 50%-100% des Messbereiches. Bei kleineren Messbereichen war er jeweils noch geringer. Geprüft wurde die Messgenauigkeit jeweils mit Probebelastungen in den einzelnen Komponentenrichtungen. Bei der Feststellung der Belastungsrichtung zeigte sich jedoch eine entgegengesetzte Tendenz. Je kleiner die zu messende Kraft war, desto ungenauer wurde ihre Richtung gemessen (Bergmann 1997b).

Allerdings konnten die Versuche insgesamt an nur drei Gelenken zweier Patienten durchgeführt werden und können deshalb nicht als repräsentativ genug erachtet werden (Bergmann 1997a).

In verschiedenen Untersuchungen konnte die mechanische, elektrische und biologische Sicherheit der Versuchsapparatur mitsamt der Telemetrie für die Versuchspersonen festgestellt werden (Bergmann 1997d).

2.4 FE-Programme

Aufgrund der Tatsache, dass die FEM schon Mitte des 20. Jahrhunderts entwickelt wurde und wenige Jahrzehnte später auch erste Rechner ihren Weg in die Forschung fanden, wurden bis heute schon sehr viele FE-Programme von verschiedenen Firmen entwickelt. Sie bieten grundsätzlich die Möglichkeit, die Qualität von Konstruktionen, in physikalischer Hinsicht, rechnerisch statt experimentell, im Voraus zu überprüfen. Das erspart bei industriellen Produktionsprozessen viele Kosten, da viele Versuche mit den Prototypen in Wirklichkeit nicht mehr durchgeführt werden müssen.

Prinzipiell gibt es FE-Programme mit kostenloser Lizenz und kostenpflichtiger Lizenz. Die kostenpflichtigen Programme haben gegenüber den Kostenlosen den Vorteil, dass sie im Gleichungslöser mit effektiveren mathematischen Verfahren arbeiten (Mathiak 2010). Des

[4] meist postoperativ nach Implantierung der Messprothese in die Versuchsperson

Weiteren verfügen sie über bessere Grafiken, haben eine höhere Grenze bezüglich der zu berechnenden Knoten und sie sind imstande mehrere physikalische Größen, außer mechanische zu berechnen.

Zu den bekanntesten kostenpflichtigen FE-Programmen zählen die Programme ABAQUS[5], ANSYS, ALGOR[6] und NEi Nastran[7]. Das bekannteste kostenlose Programm ist Z88[8]. Einige der besonderen Eigenschaften dieser Programme können der Tabelle 1 entnommen werden.

Tabelle 1: FE-Programme

FE-Programm	**Eigenschaft**
Abaqus	sehr weite Spanne an möglichen Berechnungsgrößen, u.a. auch für gummiartige Materialien
ANSYS	wird von sehr vielen Firmen, u.a. aus der Automobilindustrie, Luft- und Raumfahrt und dem Energiesektor genutzt, ursprünglich für statische, dynamische und Hitze-Fragestellungen konzipiert
ALGOR	wird hauptsächlich in der Luft- und Raumfahrt verwendet und u.a. für Kunststoffe, Metalle, Glas und Beton
NEi Nastran	ursprünglich entwickelt um den Bedarf der NASA zu decken, Berechnung von Kompositen und Beton
Z88	u.a. Berechnung der mechanischen Eigenschaft von Holz und Glas

Da in dieser Arbeit hauptsächlich mit dem FE-Programm ANSYS in der 14.0 Version gearbeitet wurde, soll rückblickend eine Schilderung der Arbeit mit diesem Programm erfolgen.

[5] Firma Dassault Systémes/Simulia, Vélizy-Villacoublay, Frankreich

[6] Firma Autodesk, San Rafael, USA

[7] Firma NEi Software, Westminster, USA

[8] entwickelt von Prof. Frank Rieg und seinem Team, Universität Bayreuth, Deutschland

2.4.1 ANSYS

ANSYS besitzt eine Benutzeroberfläche namens GUI „Graphical User Interface“, über die sämtliche Befehle bezüglich der vorzunehmenden Simulationen entweder per Mausklick über das „Main Menu“ oder per Textbefehl über die „Command Prompt“ erteilt werden können. Jeder einzelne Befehl, der vom Nutzer erteilt wird, ist zum einen nicht rückgängig zu machen und wird zum anderen in einer Textdatei namens „file.txt“ im angegebenen ANSYS-Ordner abgespeichert. Die Form, in der diese Befehle in der Datei abgespeichert werden, nennt sich APDL (ANSYS Parametric Design Language). APDL ist demnach die Programmiersprache von ANSYS. Die Tatsache, dass alle Befehle, wie oben angesprochen, automatisch in der Textdatei gespeichert werden, hat den Vorteil, dass der Nutzer, beim Schreiben eines Algorithmus, Befehle, die er per Mausklick an das Programm gesendet hat, später in der Textdatei nachschauen kann und dort auch die Schreibweise des Befehls in APDL-Form vorfindet. Das erspart beim Programmieren häufig das Nachschauen einzelner Befehle in der in ANSYS integrierten Hilfe.

Technisch gesehen verfügt ANSYS über sämtliche in dieser Arbeit beschriebenen Funktionen bezüglich des ersten und erneuten Präprozessierens und ist mit den gängigen o.g. Segmentierungsprogrammen kompatibel.

Ferner gehören, neben den für diese Arbeit in Betracht kommenden mechanischen Simulationsmöglichkeiten, auch weitere Möglichkeiten, wie Multiphysics[9], fluide Simulationen, elektrische und magnetische Simulationen u.a. nach Maxwell, elektromagnetische Simulationen und auch Kopplungsprobleme – das ist die Beobachtung der Reibung und Haftung von zwei sich bewegenden Körpern aneinander – zum Betätigungsfeld dieses Programms.

Leider gibt es dennoch ein Problem, für das es zumindest in ANSYS momentan keine hinreichend zufriedenstellende Lösung gibt. Für den Fall, dass ein Nutzer eine Simulation durchführen will, bei der sich Betrag und Richtung der Kräfte während einer Simulation in einem Verlauf, der in Wirklichkeit einem Bewegungsablauf entspricht, ändern, kann dieser Verlauf in ANSYS nicht simuliert werden, sondern es kann immer nur für Einzelzustände gerechnet werden, die in Wirklichkeit einem einzigen Augenblick entsprechen.

---

[9] engl.: Berechnung mehrerer physikalischer Größen am selben Modell

3. Material und Methode

3.1 Grundlagen der FE-Rechnung

Eine präzise Übertragung des Knochenmodells und der klinischen Kraftverhältnisse an diesem Knochen in ein FE-Programm vorausgesetzt, geht es im nächsten Schritt darum, zu verstehen, was konkret in einem FE-Programm mit den erhobenen Daten passiert und wie es zu Simulationsergebnissen kommt. Man sollte wissen, dass FE-Simulationen prinzipiell, unabhängig davon, welches FE-Programm man verwendet, immer mindestens in drei aufeinanderfolgenden Hauptschritten ablaufen: Präprozessor, Gleichungslöser und Postprozessor. In den im Anhang dieser Arbeit beigefügten Algorithmen ist es so, dass sich diese vorgegebene Struktur nicht auf Anhieb erkennen lässt. Das liegt ausschließlich an den Bedingungen, die APDL beim Schreiben eines Algorithmus vorgibt und widerspricht deshalb nicht der obigen Darstellung.

Im Präprozessor geht es, wie der Terminus vermuten lässt, um die Vorbereitung der Daten, bevor gerechnet wird. Konkret bedeutet das in Fall dieser Arbeit, dass dieser Abschnitt alle Arbeiten umfasst, in denen es darum geht, das klinische Modell mit Randbedingungen und in Form eines FE-Modells im FE-Programm zur Verfügung zu stellen, unabhängig davon, ob es sich um ein im FE-Programm vom Nutzer konstruiertes Modell oder um ein mithilfe eines Segmentierungsprogramms importierten Modells handelt.

Beim Gleichungslösen geht es um die reine Berechnung des Modells. Das heißt, dass unter Berücksichtigung aller angebrachten Randbedingungen nach Abschluss aller Arbeiten des Präprozessors, mithilfe von Differentialgleichungen berechnet wird, was mit dem Knochen passiert, wenn die angebrachten Randbedingungen auf diesen wirken. Die Ergebnisse werden jedoch noch nicht visualisiert.

Erst im letzten Schritt, Postprozessor, also Nachbereitung, geht es um die visuelle Darstellung der Ergebnisse des Gleichungslösers und gegebenenfalls um die Auswertung der Ergebnisse durch den Nutzer.

Im Folgenden werden diese drei Hauptschritte ausführlicher diskutiert.

3.1.1 Präprozessor

Wie bereits erwähnt, ist das Vorliegen des klinischen Modells in Form eines sogenannten FE-Modells inklusive Randbedingungen die Voraussetzung für eine FE-Rechnung. In diesem Abschnitt soll deshalb geklärt werden, was ein FE-Modell ist und anschließend auch, wie man dessen Randbedingungen festlegen kann.

Es handelt sich bei einem FE-Modell um ein virtuelles Modell, das in der Form dem Körper entspricht, an dem die Wirkung der Randbedingungen berechnet werden soll, also in diesem Fall der Knochen. Doch ein FE-Modell ist darüber hinaus auch vernetzt. Dass es vernetzt ist bedeutet, dass das Modell, das ursprünglich als ein normaler radiologischer (Teil-)Scan oder als ein normales im FE-Programm konstruiertes Modell vorlag, nun, auch optisch sichtbar, in viele kleine Abschnitte unterteilt ist.

Diese einzelnen Abschnitte des FE-Modells werden „Elemente“ genannt und haben immer eine bestimmte Form, die vom Benutzer gewählt werden muss. Das Muster aller Elemente, die auf das ursprüngliche Modell projiziert wurde, ist das Netz. Das Vernetzen hat den Hintergrund, dass die Wirkung der Randbedingungen, die auf das Modell angebracht werden sollen, gemäß der Theorie der FE nur berechnet werden kann, wenn diese für die einzelnen Elemente berechnet werden kann. Für jedes Element muss also jeweils einmal FE-gerechnet werden. Erst die Summe der Ergebnisse der FE-Rechnung jedes einzelnen Elements führt zu einem Gesamtergebnis, das auch optisch sichtbar gemacht werden kann. Außerdem können – softwaretechnisch gesehen – Randbedingungen auch immer nur dann aufgetragen werden, wenn ein Modell als FE-Modell, also vernetzt vorliegt.

In der FE-Terminologie gibt es deshalb die Unterscheidung zwischen einem „Festkörper“ und einem „FE-Modell“. Ein Festkörper im Sinne der FEM ist ein virtuelles Modell, welches, unabhängig davon, ob es konstruiert wurde oder aus einem Scan stammt, vorliegt, ohne vernetzt zu sein (siehe Abbildung 6a). In seiner Form entspricht es dem in Wirklichkeit zu simulierenden Modell. Da es aber nicht vernetzt ist, können an diesem auch noch keine Randbedingungen angebracht werden. Ein FE-Modell hingegen ist ein Festkörper, das schon vernetzt wurde und auf dem von nun an Randbedingungen angebracht werden können (siehe Abbildung 6b). Von entscheidender Bedeutung für das weitere Vorgehen ist also das Netz mit seinen Elementen. Deswegen soll als nächstes auf Elemente und auch auf die weiteren Bestandteile des Netzes eingegangen werden.

3.1.1.1 Elemente

Ein Element ist ein geometrisches Objekt mit bestimmten physikalischen Eigenschaften, das je nach Festkörper zwei- oder dreidimensional sein kann und dessen Form vom Nutzer für das ganze FE-Modell ausgewählt wird. Es können also z.B. für einen zweidimensionalen Festkörper Quadrate oder Dreiecke gewählt werden. Für einen dreidimensionalen Festkörper kommen z.B. Kuben oder Tetraeder in Betracht (siehe Abbildung 5). Beim Vernetzen wird der Festkörper vollständig mit Elementen der gewählten Form ausgefüllt. 3-D-Festkörper werden grundsätzlich nicht nur an ihrer Oberfläche, sondern immer auch in ihrem Inneren vernetzt, also mit Elementen gefüllt. Die Feinheit eines Netzes bzw. die Anzahl der Elemente, ist entscheidend für die Genauigkeit einer Rechnung und ebenso für die Länge der Rechendauer. Ein feines Netz, also ein Netz mit sehr vielen Elementen, liefert entsprechend sehr genaue Ergebnisse, aber die Rechenzeit erhöht sich dadurch auch, weil aufgrund der höheren Zahl an Elementen häufiger kalkuliert werden muss.

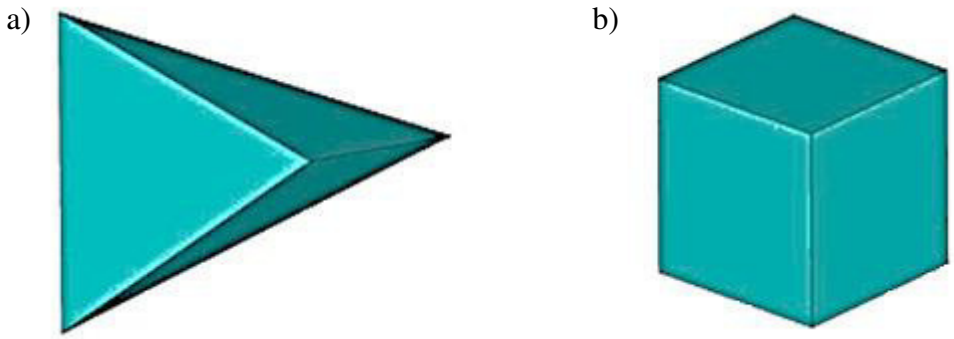


Abbildung 5: ANSYS – a) Tetraeder-Element b) Kubus-Element

3.1.1.1.1 Elementtypen

Da FE-Programme nicht nur mechanische, sondern zum Beispiel auch thermische, fluide und elektrische Strömungssimulationen durchführen können, gibt es unter den Kategorien von Elementtypen einige, die für mechanische Simulationen, wie die Fragestellung dieser Arbeit, nicht geeignet sind und andere, die besonders gut geeignet sind (Park 2013). Da in dieser Arbeit hauptsächlich mit dem FE-Programm ANSYS gearbeitet wurde, werden in der nachstehenden Tabelle die in diesem Programm zur Verfügung stehenden für mechanische Simulationen geeigneten Kategorien von Elementtypen mit ihren wichtigsten besonderen Eignungen genannt.

Tabelle 2: Kategorien von Elementtypen

Kategorien von Elementtypen	**Eigenschaften**
Beam	uniaxiales Element für Simulationen mit Spannungen, Kompressionen und Beugungen
Combination	drehgelenkartiges Element, das zur Verbindung zweier Modellabschnitte an einem bestimmten Punkt gebraucht werden kann. Es zeichnet sich durch besondere Eignungen in der Flexibilität, Friktion und in Dämpfungsversuchen aus
Contact	wird in Simulationen verwendet, in denen es darum geht, ob zwei sich bewegende voneinander unabhängige Oberflächen aneinander hängen bleiben oder ihren physikalischen Kontakt aneinander verlieren
Fluid	wird in Flüssigkeitsmodellen benutzt und im Kontaktbereich zwischen flüssigen und mechanischen/thermischen Modellen
Hyperelastic	kann in hyperelastischen Modellen genutzt werden bis hin zu gummiartigen Modellen, die fast nicht mehr zusammengedrückt werden können
Infinite	ist zur Modellierung eines grenzenlosen Feldes geeignet
Interface	wird zur Verbindung zwischen magnetischen Vektoren und skalaren Potenzialen im selben Modell genutzt
Link	kann in einer Vielzahl von Modellen verwendet werden. Abhängig von der Fragestellung kann es als Verbindungselement, Trägers oder Feder dienen
Mass	ein punktförmiges Element mit ein oder mehreren Freiheitsgraden, kann auch eine Temperatur haben
Matrix	ein arbiträres Element ohne definierte Geometrie, dessen elastische Kinematik in der Steifigkeit, in der Dämpfung und in der Masse bestimmt werden kann
Pipe	ein uniaxiales Element für Spannungskompression, Torsion, Biegung

Plane	Das Element hat eine quadratische Verlagerungseigenschaft und passt sehr gut zu Modellen mit unregelmäßigen Netzen, wie zum Beispiel Modelle, die aus CAD/CAM stammen.
Shell	wird vorzugsweise bei Scherspannungen gebraucht
Solid	hat herausragende Fähigkeiten bei dreidimensionalen magnetischen, thermalen, elektrischen, piezoelektrischen und strukturellen Feldproblematiken
Surface	kann für verschiedene Kraft- und Oberflächeneffekte gebraucht werden
Target	wird zur Darstellung von 2-D Zieloberflächen für einige Contact-Elemente benutzt
Viscosity	wird für die 2-D-Modellierung von soliden Strukturen genutzt

Innerhalb einer Kategorie von Elementtypen gibt es wiederum je nach Programm unterschiedliche Anzahlen von verschiedenen Elementtypen. Zum Beispiel gibt es im FE-Programm ANSYS in der Plane-Kategorie 17 verschiedene Elementtypen, bei ABAQUS weit über 60. Verschiedene Elementtypen derselben Kategorie können sich in ihrer Dimensionszahl – es gibt prinzipiell zwei- und dreidimensionale Elemente -, in der Anzahl ihrer Knoten – auf diesen Begriff soll in Kapitel 3.1.1.2 genauer eingegangen werden -, in ihrer Form und in weiteren physikalischen Eigenschaften unterscheiden.

Ein FE-Modell kann jedoch immer nur aus Elementen desselben Typs bestehen. Möchte man im selben Modell mehrere Elementtypen haben, so lässt sich in manchen FE-Programmen die Option nutzen, mehrere Modelle, von denen jedes aus Elementen anderen Typs oder anderer Kategorien besteht, zu koppeln. Für die Fragestellung dieser Arbeit scheinen sich, von den zur Verfügung stehenden Elementen, diejenigen aus der PLANE-Kategorie am besten zu eignen, da besonders bei der Fragestellung dieser Arbeit mit besonders unregelmäßigen Formen gerechnet werden muss und diese auch möglicherweise auf CAD/CAM basieren.

3.1.1.2 Knoten

Abgesehen von Elementen sind auch die Knoten feste Bestandteile von FE-Modellen (Mathiak 2010). Diese sind Punkte, die über das gesamte FE-Modell mehr oder weniger gleichmäßig verteilt sind und die maximal sechs Freiheitsgrade besitzen können. Abbildung 6d zeigt beispielsweise dasselbe FE-Modell, das in Abbildung 6c dargestellt ist; es sind aber lediglich die Knoten des FE-Modells eingeblendet. Man sieht, dass es sich dabei um ein FE-Modell mit Tetraeder-Elementen handelt, die jeder nur an der Ecke einen Knoten aufweisen. Je nach Elementtyp befinden sich Knoten an den Ecken eines Elements, in der Mitte der Seiten eines Elements, in den Mittelpunkten der Flächen von Elements, im Mittelpunkt des Volumens dreidimensionaler Elementen, oder auch an anderen Stellen. Alle Kombinationen sind möglich. An welchen Stellen ein Element eines bestimmten Typs seine Knoten hat, ist immer schon im jeweiligen Programm vordefiniert. Drei der sechs Freiheitsgrade, die ein Knoten besitzen kann, sind räumlicher Natur. Ein Knoten ist in einem Raum durch eine x-, eine y- und eine z-Koordinate definiert und jede dieser Koordinatenachsen steht für einen Freiheitsgrad. Die drei weiteren Freiheitsgrade beziehen sich auf die Rotation des Knotens um die drei Raumachsen.

Theoretisch gesehen machen Rotationen von Knoten zwar keinen Sinn, weil dann mit dem Punkt an sich nichts passieren wird. Wenn man sich aber vorstellt, dass später die besagten Randbedingungen an einem Knoten ansetzen werden, so kann mit der Rotation des Knotens beispielsweise sehr einfach auch eine Rotation einer auf diesem Knoten anliegenden Kraft erwirkt werden, z.B. um eine nicht-orthogonal anliegende Kraft am Modell zu simulieren. Gäbe es diese Option nicht, so müsste man, um eine nicht genau in x-, y- oder z-Richtung ansetzende Kraft – zum Beispiel eine Kraft von 1 Newton, die von der positiven x-Achse um 30° in Richtung der positiven y-Achse abweicht – am Knoten anzubringen, im Programm ANSYS z.B. erst mithilfe der Vektoraddition zwei repräsentative Kräfte berechnen, deren Vektorsumme dann dem einen Newton entspricht, das man simulieren möchte. Mit der Drehung von Knoten kann man diesen komplizierten Zwischenschritt also umgehen.

Die Rotation eines Knotens um die x-Achse verläuft in der y-z-Ebene, die Rotation um die y-Achse in der x-z-Ebene und die Rotation um die z-Achse in der x-y-Ebene. Entsprechend hat ein Knoten, der sich in einem nur zweidimensionalen System befindet, nur maximal vier Freiheitsgrade, weil ein räumlicher und auch ein rotatorischer Freiheitsgrad fehlen.

Knoten sind in FE-Modellen die einzigen Stellen, an denen sämtliche Randbedingungen angebracht werden können.

Im FE-Programm ANSYS beispielsweise gibt es in der Plane-Kategorie den Elementtyp PLANE2. Es handelt sich dabei um ein Dreieck, das durch sechs Knoten definiert wird. Drei dieser Knoten liegen an den Ecken des Elements und drei weitere liegen jeweils in den Mitten

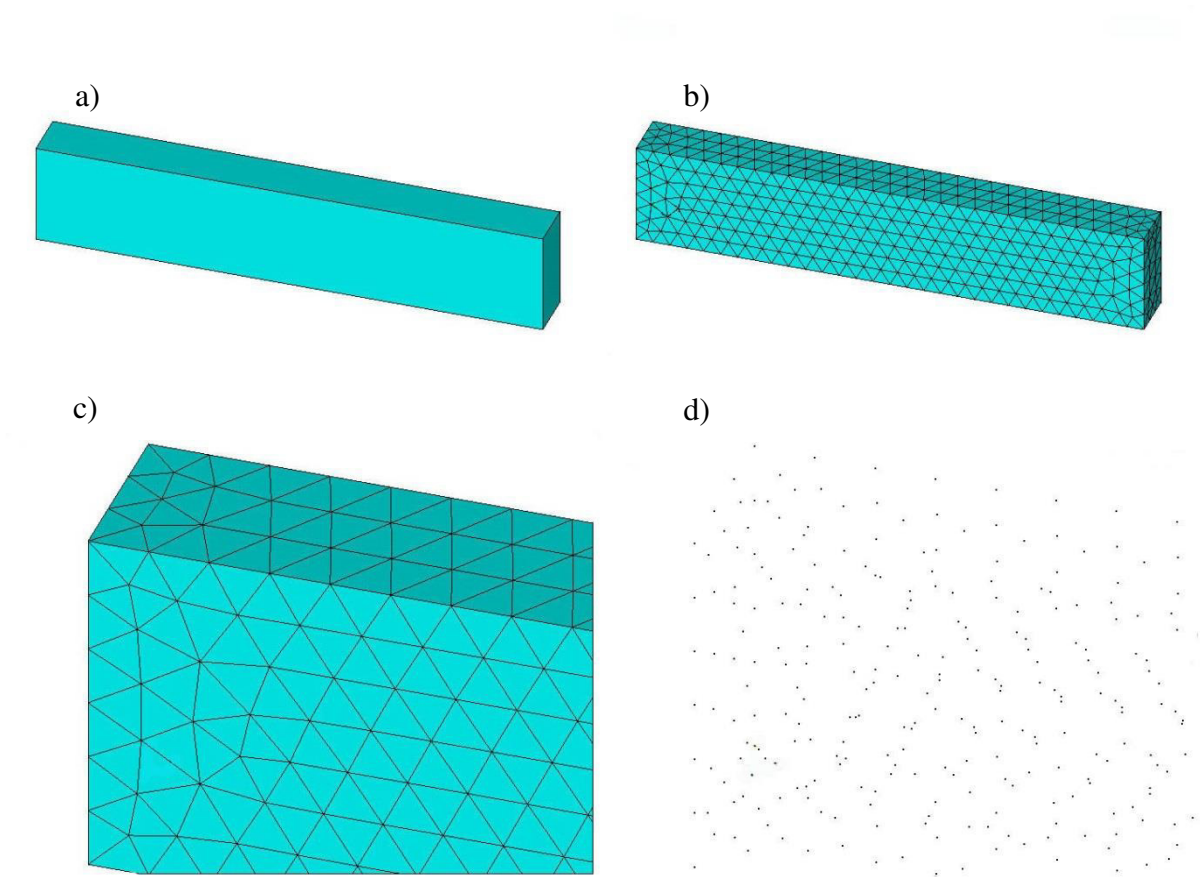


Abbildung 6: ANSYS – a) Das Balken-Modell liegt als Festkörper vor b) dasselbe Balken-Modell ist vernetzt und liegt nun als FE-Modell vor c) Vergrößerung des FE-Modells aus Abbildung 6b d) Ausblendung aller Elemente und Einblendung der Knoten der Abbildung 6c

der Seiten des Elements. Wenn man sich nun vorstellt dass zwei PLANE2-Elemente jeweils mit einer Seite aneinander liegen, so haben die beiden Elemente insgesamt keine zwölf Knoten, sondern nur neun, denn sich überlappende Knoten zählen nur einfach. Es gibt also an der Seite, an der die beiden Dreiecke aneinander liegen, insgesamt nur drei Knoten (siehe Abbildung 7).

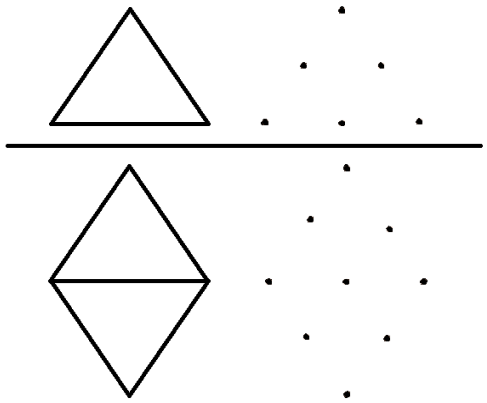


Abbildung 7: aneinander liegende Dreieck-Elemente teilen sich gemeinsame Knoten

Für die eigentliche Simulation im Sinne der FEM, also der Berechnung mithilfe von Differentialgleichungen, spielen die Knoten die entscheidende Rolle. Auf diese Tatsache soll in Kapitel 3.1.2.2 genauer eingegangen werden.

3.1.1.3 Ankerpunkte

Wenn man annimmt, dass Randbedingungen bei FE-Modellen nur an Knoten angebracht werden können, dann ist es für den Benutzer zum einen sehr wichtig, festlegen zu können, wo diese Knoten liegen, um so auch bestimmen zu können, wo die Randbedingungen am Modell liegen sollen und zum anderen sichergestellt zu haben, dass diese Knoten sich während einer Simulation nicht von der Stelle bewegen. Grundsätzlich kann der Benutzer nämlich beim Erstellen des Netzes einige Parameter, wie die Feinheit oder die Elementtypen wählen. Doch das definitive und exakte Muster des Netzes und damit auch die Lokalisation der einzelnen Knoten lassen sich nicht durch den Nutzer bestimmen, sondern es wird automatisch erstellt. Aufgrund dieser Tatsache gibt es sogenannte Ankerpunkte, deren Position im Festkörper, vor Erstellung des Netzes, vom Benutzer frei gewählt werden kann. Ankerpunkte sind also Punkte, ähnlich wie Knoten, mit dem Unterschied, dass das FE-Programm sich bei der Erstellung oder Änderung des Netzes an ihnen orientiert (Müller 2007). Zwar kann der Benutzer keine Randbedingungen an Ankerpunkten anbringen, doch das FE-Programm sorgt, aufgrund der Positionen der festgelegten Ankerpunkte dafür, dass jederzeit an den Stellen, wo

Ankerpunkte sind, auch Knoten vorhanden sind, an denen Randbedingungen angebracht sein können. Aufgrund der Tatsache, dass Ankerpunkte sich nicht bewegen können und durch die Bestimmung der Ankerpunkte auch sichergestellt ist, dass sich immer hier ein Knoten befinden muss, kann der Nutzer davon ausgehen, dass sich seine Randbedingungen immer an der korrekten Stelle befinden.

3.1.1.4 Randbedingungen

Da es, wie in Kapitel 5.1 erwähnt, bis jetzt keine Methode gibt, mit der die vorherrschenden Randbedingungen eines klinischen Modells automatisch in ein FE-Programm übertragen werden können, muss nach dem jetzigen Stand der Dinge davon ausgegangen werden, dass es vorerst weiterhin dem Benutzer obliegt, diese manuell in das FE-Programm einzugeben. Bei den Randbedingungen handelt es sich um Kräfte und Lager (Müller 2007). Was im Hinblick auf die Anwendung dieser am FE-Modell zu beachten ist, wird als nächstes diskutiert.

3.1.1.4.1 Kräfte

Kräfte, visuell meist durch Vektorpfeile symbolisiert, lassen sich prinzipiell immer auf Knoten anbringen (siehe Abbildung 8). Der Betrag einer Kraft kann vom Nutzer gewählt werden und hat die Einheit Newton. Negative Kräfte bedeuten eine Kraft in die Gegenrichtung. Die Richtung einer Kraft kann, wenn sie nicht entlang einer Achse des Koordinatensystems verläuft, prinzipiell entweder durch das Ersetzen dieser Kraft mithilfe mehrerer durch Vektoraddition berechneter repräsentativer Kräfte, die alle an demselben Knoten anliegen, oder einfach durch die Drehung des Knotens berechnet werden. Die Drehung des Knotens um bestimmte Grade bewirkt gleichzeitig eine Drehung der an ihm anliegenden Kraft. Die Drehung eines Knotens hat für die Struktur des Netzes aber keinerlei Folgen.

3.1.1.4.2 Lager

Grundsätzlich sollte man wissen, dass es für eine Simulation immer notwendig ist, auch Lagerstellen am Modell anzubringen. Diese können, genau wie Kräfte, auch immer nur an Knoten angebracht werden (siehe Abbildung 8). Verwendet man in einem Modell keine Lagerstellen und bringt lediglich Kräfte an, so kann es passieren, dass die Simulation zu keinem Ergebnis führt, da das Modell ja sozusagen aufgrund der fehlenden Lagerstellen nicht

festgehalten wurde und die angebrachte Kraft dadurch nicht wirken konnte, weil sie das Modell nur vor sich her geschoben hat.

Beim Anbringen von Lagerstellen ist zu beachten, dass an diesen je nachdem, ob man sich in einem 2-D- oder einem 3-D-System befindet, einzelne Freiheitsgrade gesperrt werden können. Man kann ein Lager an einem 2-D-Modell an einem Knoten also z.B. in x-Richtung sperren und in y-Richtung frei lassen. Somit würde man der angebrachten Kraft erlauben, diesen Knoten in y-Richtung zu verschieben, jedoch nicht in x-Richtung. Analog dazu sind auch alle anderen Möglichkeiten in 2-D- und in 3-D-Systemen gegeben.

3.1.2 Gleichungslöser

Nach Abschluss des gesamten Präprozessierens, also sobald ein FE-Modell mit allen Randbedingungen im FE-Programm vorliegt, kann der Gleichungslöser gestartet werden. In diesem Kapitel soll es nun darum gehen, was bei der eigentlichen FE-Rechnung passiert, wie es also zu einem Ergebnis kommt.

Bevor jedoch mit dem eigentlichen Rechnen begonnen wird, durchläuft der Gleichungslöser am FE-Modell Plausibilitätstests, welche überprüfen, ob das Netz des FE-Modells für eine Rechnung geeignet ist, oder ob möglicherweise einzelne Elemente so stark verzerrt sind, dass nicht gerechnet werden kann. Im letzteren Fall wird nicht gerechnet, sondern es erscheint eine Fehlermeldung, die auf dieses Problem hinweist. Um dieses Problem zu lösen, muss das Netz besser konstruiert werden. Zum anderen wird in den Plausibilitätstests geprüft, ob alle Elemente, an deren Knoten Randbedingungen angebracht sind, noch räumlich miteinander zusammenhängen. Eine Kraft oder ein Lager an einem Knoten, dessen Element durch eine Lücke von allen anderen Elementen getrennt ist, ist nicht sinnvoll, weil sie technisch nicht wirken kann und der Nutzer wird deshalb auf dieses Problem hingewiesen (siehe Abbildung 8).

Sollten die Plausibilitätstests jedoch ergeben, dass das Netz zum Rechnen geeignet ist, so erscheint weiter keine Meldung und es wird gerechnet. Ein grober Überblick, der nicht den Rahmen dieser Arbeit sprengen soll, darüber, was die FEM ist und wie gerechnet wird, soll im nächsten Kapitel vermittelt werden.

3.1.2.1 Was ist die FE-Methode?

Die FEM ist ein mathematischer Ansatz, der zum Berechnen physikalischer Simulationen, wie z.B. mechanischen Spannungen, innerhalb eines vorgegebenen Körpers, unter den jeweils vorgegebenen Randbedingungen (Mathiak 2010). Sie setzt, unabhängig davon, welche physikalische Größe berechnet werden soll, voraus, dass das Werkstück als FE-Modell vorliegt. Für jedes der einzelnen Elemente des FE-Modells wird je über eine eigene Differentialgleichung unter Berücksichtigung der am gesamten Modell anliegenden Randbedingungen die gefragte physikalische Größe berechnet (Mathiak 2010).

Die Berechnung der gefragten physikalischen Größe für jedes einzelne Element ergibt zum Schluss eine Gesamtbilanz für das gesamte Modell, die im Postprozessor auch graphisch dargestellt werden kann. Dort kann der Nutzer für jeden Bereich des Modells Rückschlüsse auf die Wirkungen der Randbedingungen ziehen. In der Wirklichkeit besteht ein Festkörper – und damit auch ein Knochen, anders als in der FEM angenommen, aus einem soliden, zusammenhängenden und nicht unterteilten Werkstück, in dem sich die Kraft, kontinuierlich und nicht abschnittweise, über jede Stelle ausbreiten kann. Da Festkörpersimulationen mit dieser Annahme jedoch prinzipiell nicht berechnet werden können, stellt die FEM, mit dem Ansatz der Unterteilung des Werkstücks in eine bestimmte, steigerungsfähige Anzahl von Elementen, die beste bisher bekannte Methode dar, mit der solche Simulationen berechnet werden können.

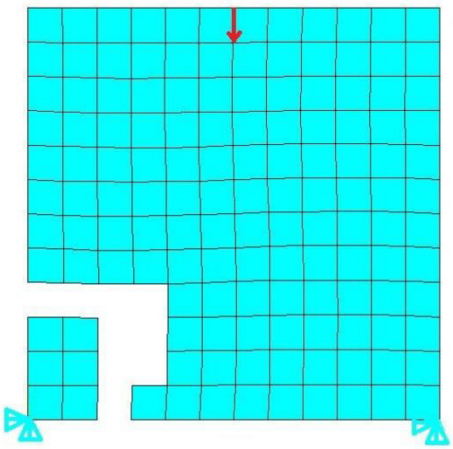


Abbildung 8: ANSYS – roter Pfeil = Kraft, blaue Keile = Lager in x- und y-Richtung; das Modell besteht die Plausibilitätskontrolle aufgrund der fehlenden Elemente im linken unteren Bereich nicht, weil dadurch das Modell nicht genügend abgestützt ist.

3.1.2.2 Wie funktioniert die FE-Rechnung?

In diesem Kapitel wird ein grober Überblick darüber gegeben werden, wie man anhand der FEM am Beispiel einer Spannungssimulation zu einem Ergebnis kommt.

Wenn auf einen Körper geringe Kräfte wirken, spricht man bei diesem Körper von einem linear-elastischen Verformungsverhalten. Auf dieses Verformungsverhalten ist das Hookesche Gesetz

$$D = \frac{F}{U} \tag{1}$$

anzuwenden, welches ursprünglich zur Berechnung von Auswirkungen an Federn formuliert wurde. Es besagt, dass die an der Feder wirkende Kraft F und die dadurch entstehende Auslenkung U der Feder über die Konstante D proportional zueinander stehen (Nasdala 2015).

An FE-Modellen muss eine für das Gesamtmodell zu berechnende Verformung immer erst einzeln für jedes Element berechnet werden, bevor danach aus den einzelnen Ergebnissen ein Gesamtergebnis errechnet wird. Das Hookesche Gesetz angewandt an einem einzelnen beispielsweise durch zwei Knoten ($N1$ und $N2$, beide auf der x-Achse gelegen) festgelegten Element, auf das die Kraft genau entlang der Länge des Elements wirkt, hieße, dass die wirkende Kraft F entspricht, U wäre die durch die Kraft F bewirkte Längenänderung des Elements. Die Konstante D, welche bei Federn der Federkonstante entspricht, wäre in diesem Fall der sogenannte Elastizitätsmodul E des Elements. Es kommt durch die wirkende Kraft also zu einer Abstandsänderung der beiden Knoten des Elements zueinander, welche man auch als Längenänderung dieses Elements interpretieren kann und die umso größer ist, je größer die wirkende Kraft auch ist.

Da bei Festkörpersimulationen, im Gegensatz zu Federsimulationen, nie eine Querschnittsfläche A von null für das Element angenommen werden kann, muss nun auch für die Berechnung der Verformung in unserem Beispiel von einer bestimmten Querschnittsfläche A des Elements ausgegangen werden. Der Zusammenhang zwischen der Federkonstante D, dem Elastizitätsmodul E und der Elementlänge L ergibt sich über folgende Formel (Nasdala 2015):

$$D = \frac{E \cdot A}{L} \tag{2}$$

Insgesamt entsteht als Ergebnis aus den Formeln (1) und (2) folgender Zusammenhang für die bewirkte Längenänderung:

$$U = \frac{F \cdot L}{E \cdot A} \tag{3}$$

Möchte man mehrdimensionale Festkörpersimulationen berechnen, so muss man diese Formel zur Berechnung der Verschiebung jedes einzelnen Knotens und zwar einzeln entlang jeder Dimensionsachse anwenden. Unter der Annahme, dass die in diesem Beispiel wirkende Kraft genau in Richtung des Elements, ansetzend an N1 wirkt, ergeben sich zur Berechnung der wirkenden Kraft an N1 jeweils entlang der Koordinatenachse x folgende Gleichungen, sofern N2 starr ist (Nasdala 2015):

$$F_{N1} = \frac{E \cdot A}{L} \cdot (u_1) \tag{4}$$

$$F_{N2} = \frac{E \cdot A}{L} \cdot (-u_1) \tag{5}$$

Nimmt man nun an, dass N1 starr ist und N2 beweglich, so ergeben sie dafür entsprechend die Formeln:

$$F_{N1} = \frac{E \cdot A}{L} \cdot (u_2) \tag{6}$$

$$F_{N2} = \frac{E \cdot A}{L} \cdot (-u_2) \tag{7}$$

Für den Fall, dass sowohl N1, als auch N2 beweglich sind, lassen sich diese Formeln in Bezug auf das ganze Element auch als folgende Gleichungen darstellen, die auch die Elementsteifigkeitsmatrix $[K]_e$ genannt wird, ausdrücken:

$$\begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} \frac{E \cdot A}{L} & -\frac{E \cdot A}{L} \\ -\frac{E \cdot A}{L} & \frac{E \cdot A}{L} \end{pmatrix} \cdot \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{8}$$

$$\vec{F} = K_e \cdot \vec{u} \tag{9}$$

Die aufgestellte Formel gilt nun für den Fall, dass die Knoten des zu berechnenden Elements genau in der x-Achse liegen. Sollte dies nicht der Fall sein, muss in dieser Formel mit den entsprechenden Sinus- und Cosinus-Werten gerechnet werden.

Hat man für alle Elemente die jeweilige Elementsteifigkeitsmatrix K_e zu einer einzigen sogenannten Gesamtsteifigkeitsmatrix K_{ges} zusammengefügt, können unter Berücksichtigung der gegebenen Randbedingungen $\vec{F}$ die Verschiebungen der einzelnen Knoten $\vec{u}$ entlang jeder Koordinatenachse ausgerechnet werden. Wichtig ist, dass zumindest für einige Knoten des Modells vorher bestimmte Freiheitsgrade gesperrt wurden. Denn ein Modell, bei dem sämtliche Knoten in allen Freiheitsgraden frei gelagert sind, wird von einer Kraft, die auf das Modell wirkt, lediglich vor sich her geschoben. Eine Kraft kann also nur auf ein Modell wirken, bei dem zumindest ein Knoten in Richtung der Kraft gesperrt ist. Die Gesamtsteifigkeitsmatrix wird in FE-Programmen automatisch ausgerechnet und kann je nach Modellgröße und Anzahl der Freiheitsgrade mehrere Millionen Spalten und Zeilen enthalten. Zum Errechnen der gesuchten Größe $\vec{u}$ sind FE-Programme mit besonderen Algorithmen ausgestattet, die mit Differentialgleichungen arbeiten und prinzipiell dazu ausgelegt sind mit geringem Rechenaufwand die gesuchten Ergebnisse zu ermitteln.

Die hergeleitete Formel (9) gilt allerdings, wie bereits erwähnt, für sogenannte linear-elastische Analysen. Dabei wird davon ausgegangen, dass die Steifigkeit der einzelnen Elemente sich während ihrer Verformung nicht ändert, wie am Beispiel der Feder. Da bei Knochensimulationen, wie Frost zeigen konnte (Frost 1987), davon auszugehen ist, dass sich die einzelnen Elemente, um der Realität vitalen Knochens zu entsprechen, in ihrer Steifigkeit während der Verformung ändern, muss auch davon ausgegangen werden, dass Knochensimulationen in der FEM nichtlinear-elastisch ablaufen. Deswegen muss zu der bereits aufgestellten Gesamtsteifigkeitsmatrix noch die Steifigkeitsmatrix für nichtlineare Verformungen summiert werden.

Im nächsten Schritt, also im Postprozessor, können die vom Benutzer gefragten Größen, wie z.B. die Druck- oder Zugspannung, aufgrund der vorangegangenen Kalkulation graphisch dargestellt werden (siehe Abbildung 9). Das funktioniert in Bezug auf Zug- und Drucksimulationen auf die Weise, dass immer dort, wo sich zwei Knoten aufeinander zu verschoben haben, ein Druck verzeichnet wird und dort, wo sie sich voneinander entfernt haben, Zug. Der Druck und der Zug sind jeweils umso größer, je intensiver die Verschiebung war.

Zusammenfassend ist es wichtig zu wissen, dass die später graphisch dargestellten Spannungen aufgrund der mathematisch berechneten räumlichen Verschiebung der einzelnen Knoten berechnet wurden.

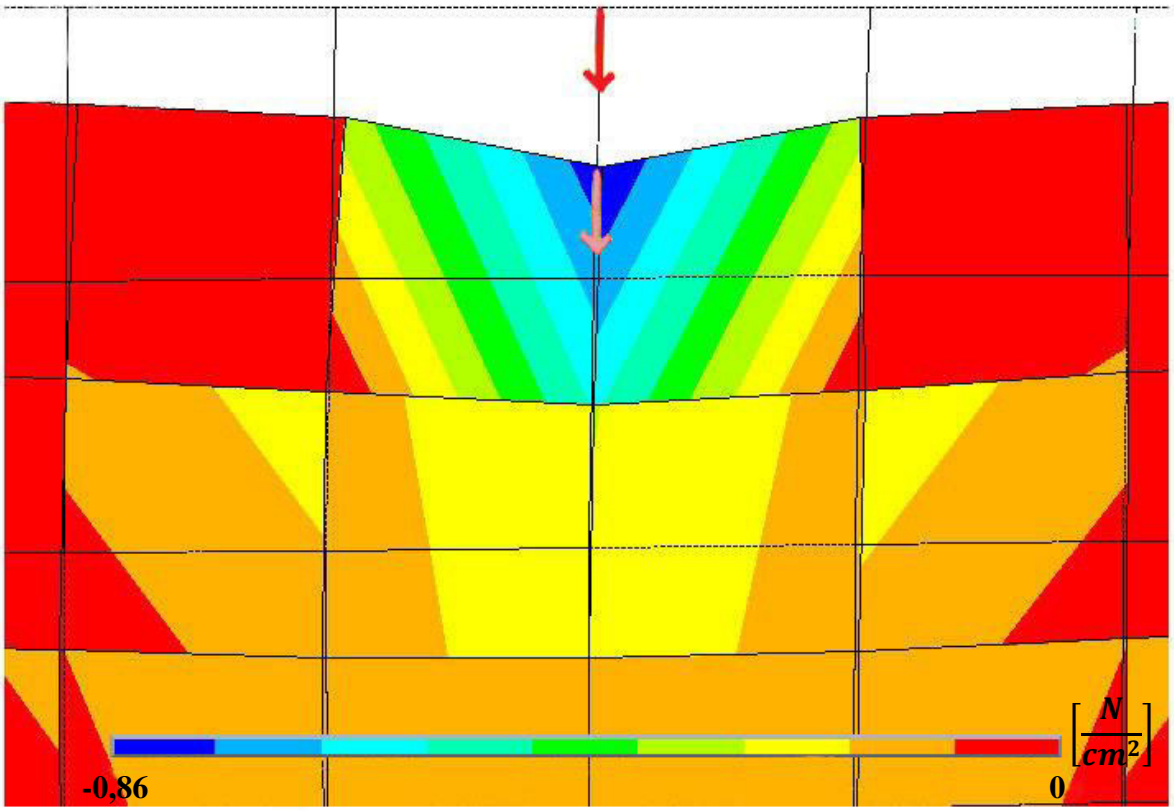


Abbildung 9: ANSYS – Ausschnitt aus der Drucksimulation eines FE-Modells, Kraft 1 N (dunkelroter Pfeil); man sieht durch die Überlagerung des unverformten Ausgangsmusters des FE-Modells und des verformten FE-Modells die bewirkten Knotenverschiebungen und die Verformung der Elemente (Vorher/Nachher-Vergleich)

3.1.3 Postprozessor

Im Postprozessor werden die im Gleichungslöser berechneten Ergebnisse visualisiert (Fröhlich 1995). Hierfür gibt es sehr viele verschiedene Optionen. Bevor der Benutzer sich entscheidet, bezogen auf mechanische Simulationen, welche der vielen verschiedenen Arten von Spannungen, Druck- und Zugverhältnisse er angezeigt haben möchte, steht zunächst grundsätzlich die Entscheidung an, ob man seine Ergebnisse im elementbasierten Lösungsansatz oder im knotenbasierten Lösungsansatz angezeigt haben möchte (Madenci

2015). Diese beiden Ansätze heißen in ANSYS „Element Solution“[10] und „Nodal Solution“[11]. Sie unterscheiden sich darin, dass in der elementbasierten Lösung die angezeigten Spannungen Element für Element aufgrund der räumlichen Verschiebung der einzelnen Knoten berechnet werden. Die knotenbasierten Lösung hingegen basiert auf den Mittelwerten der Ergebnisse der elementbasierten Lösung, die für das gesamte Modell berechnet werden.

3.1.3.1 Elementbasierte Lösung

Bei der elementbasierten Lösung werden die Ergebnisse so angezeigt, wie sie in der oben erwähnten Verschiebungsfunktion, also aufgrund der räumlichen Verschiebung der einzelnen Knoten, Element für Element berechnet wurden (Awang 2016). Da diese Verschiebungen nicht immer zwangsweise linear verlaufen, kann das manchmal dazu führen, dass es in der Darstellung von Spannungen im Modell zu Inhomogenitäten kommt (siehe Abbildung 10b). Elemente, oder Areale in Elementen mit hohen Spannungen liegen manchmal unmittelbar neben Elementen, oder Arealen von Elementen, in denen deutlich geringere Spannungen liegen. Das erscheint auf den ersten Blick nicht plausibel, ist aber dadurch zu erklären, dass es in der elementbasierten Lösung eben nur die einzelnen Ergebnisse jedes Elements lediglich nebeneinander dargestellt werden. Das Programm prüft also nicht die Plausibilität der Ergebnisse der einzelnen Elemente in Bezug auf einander. Es ist, wie vorher gesagt, auch nicht zu erwarten, dass sich die Knoten immer linear verschieben.

3.1.3.2 Knotenbasierte Lösung

Im Gegensatz zur elementbasierten Lösung werden bei der knotenbasierten Lösung nicht Einzelergebnisse angezeigt. Stattdessen werden, ausgehend von den Ergebnissen der elementbasierten Lösung, an den einzelnen Knoten jeweils die Mittelwerte der Ergebnisse der anliegenden Elemente angezeigt (Awang 2016). Das sorgt graphisch für einen weicheren Spannungsverlauf und macht auf den Nutzer einen plausibleren Eindruck (siehe Abbildung 10a). Während in der elementbasierten Lösung die Ergebnisse streng nach der Theorie der FE

[10] engl.: elementbasierte Lösung

[11] engl.: knotenbasierte Lösung

angezeigt werden, also mit der Annahme, Element für Element Ergebnisse zu berechnen und anzuzeigen, entsteht beim Betrachten von knotenbasierten Lösungen der Eindruck, dass die Ergebnisse so angezeigt werden, wie sie in Wirklichkeit auch vorliegen, also durchgehend und gleichmäßig verteilt durch das gesamte Modell, als wäre das Modell gar nicht in einzelne Elemente unterteilt, sondern zusammenhängend. Trotzdem darf sich der Nutzer von der knotenbasierten Lösung nicht dazu verleiten lassen zu denken, dass es sich hierbei um die „echten“ Ergebnisse handele. Diese sind nach wie vor in der elementbasierten Lösung zu finden.

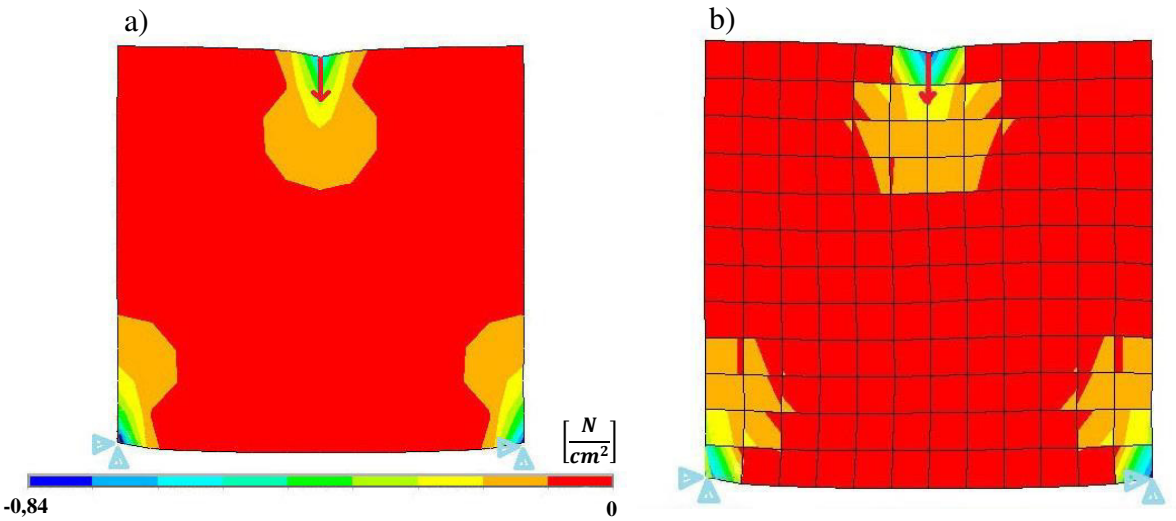


Abbildung 10: ANSYS – roter Pfeil = Kraft (1 N), blaue Keile = Lager in x- und y-Richtung, Modell-Größe 10x10cm; die gleiche Drucksimulation eines FE-Modells, a) knotenbasierte Lösung b) elementbasierte Lösung

3.1.3.3 Spannungen

Nachdem der Nutzer sich für eine der beiden Optionen, knotenbasierte oder elementbasierte Lösung entschieden hat, steht als nächstes die Frage an, welche Art von Spannung überhaupt angezeigt werden soll. Anhand der Ergebnisse der Verschiebungsfunktion aus dem Gleichungslöser können viele verschiedene Arten von Spannung, wie Druck-, Zug- und Von-Mises angezeigt werden.

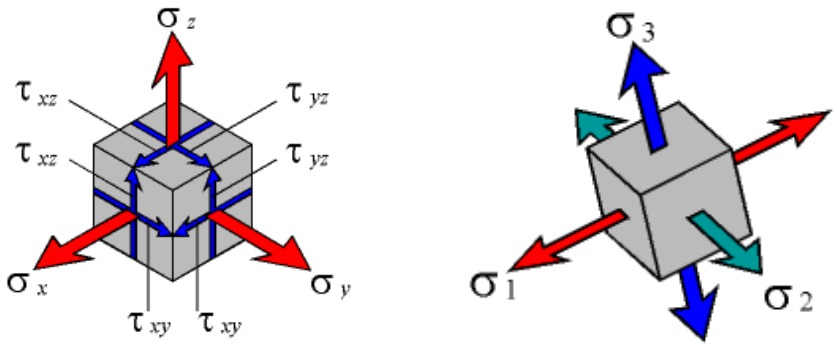


Abbildung 11: Die Spannungen in deren nominalen Richtungen, σ_x, σ_y und σ_z, und die drei Scherspannungen, τ_{xy}, τ_{yz} und τ_{xz}.

Die Spannung Von Mises ist wie folgt definiert:

$$\sigma_v = \sqrt{\frac{(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2}{2}} \tag{10}$$

wobei 1, 2 und 3 die Hauptrichtungen sind.

Ein FE-Modell wird so gedreht, dass nur Normalspannungen verbleiben und alle Scherspannungen Null sind.

Die Hauptspannungen werden immer so geordnet, dass $\sigma_1 > \sigma_2 > \sigma_3$.

Im FE-Programm ANSYS können, abgesehen von für diese Arbeit uninteressanten Größen wie die thermische Entwicklungen in einem FE-Modell, elektrischen Kraftfeldern und Strömen, insgesamt 14 verschiedene Arten von mechanischen Spannungen angezeigt werden. Für eine zielführende Simulation ist es wichtig, zu wissen, welche dieser 14 Arten von mechanischen Spannungen am besten für die Simulation von Knochen geeignet ist. Diese Frage ist jedoch für die Entwicklung eines Algorithmus nicht entscheidend, da in einem Algorithmus die ausgesuchte anzuzeigende Spannung mit minimalem Aufwand auch vom Nutzer geändert werden kann. Trotzdem sollen die Spannungen, die am ehesten für die Berechnung von Knochen in Betracht kommen, kurz umrissen werden. Dem geht die Überlegung und die Frage voraus, ob für Ab-, Auf-, und Umbau im Knochen, die Zugspannungen, die Druckspannung oder beide zusammen verantwortlich sind. Diese Frage

wurde in der Literatur häufig diskutiert und es lassen sich teilweise gegensätzliche Ansichten finden. Während nach Wolff gemeinhin angenommen wird, dass bei Zugbelastung die Knochenmasse zunimmt (Wolff 1892), stellte hingegen Watson-Jones fest, dass bei Druck die Stabilität des Knochens erhöht wird (Watson-Jones 1955). Auch Wagner beobachtete in einem Experiment bei Druck eine Verdichtung des Knochens (Wagner 1960). Im Rahmen dieser Arbeit stellte es sich nicht als notwendig heraus, eine dieser beiden konträren Thesen als Grundlage dieser Arbeit zu nehmen. Das liegt daran, dass, wie weiter oben schon erwähnt, es in dieser Arbeit um die Entwicklung eines Algorithmus ging und dieser Algorithmus kann mit einer kleinen Änderung so umfunktioniert werden, dass er statt mit Zugspannungen mit Druckspannungen rechnet und umgekehrt. Aus diesem Grund sollen als nächstes die Rollen der Spannung erster Ordnung[12], zweiter Ordnung[13] und dritter Ordnung[14], die in beiden Thesen repräsentiert sind, diskutiert werden.

Die Spannung erster Ordnung ist reine Zugspannung. D.h., dass für jedes Element des FE-Modells lediglich berechnet wird, wie groß der Zug ist, der auf dieses Element wirkt. Der Druck wird gar nicht berechnet. Sollte für bestimmte Elemente faktisch trotzdem eine Drucksituation vorliegen und kein Zug, so werden diese Elemente mit einem Zug mit dem Betrag null angegeben (Waguespack 2009). Zugspannungen werden in ANSYS prinzipiell immer mit positiven Werten gekennzeichnet. Das heißt, dass bei einer Berechnung der Spannung erster Ordnung die Spannungswerte der Elemente bei null anfangen und ins positive gehen und daher keine negativen Werte enthalten.

Die Spannung zweiter Ordnung ist eine Kombination aus Zug- und Druckspannung. Hier wird also für jedes Element die „echte“ Situation wiedergegeben. In dieser Berechnung sind sowohl Elemente, die unter Zug stehen und positive Werte haben, als auch welche, die unter Druck stehen und deshalb negative Werte haben.

Schließlich gibt es noch die Spannung dritter Ordnung, bei der nur Druck-Werte berechnet werden. Analog zur Spannung erster Ordnung werden hier alle Elemente mit einem negativen

[12] engl.: 1st principal stress

[13] engl.: 2nd principal stress

[14] engl.: 3rd principal stress

Wert gekennzeichnet und sollten sie doch unter Zug stehen, so haben sie wiederum einen Nullwert. Positive Werte gibt es nicht (Waguespack 2009).

3.1.4 Erneutes Präprozessieren

Nachdem der erste Präprozessor, der Gleichungslöser und der Postprozessor einmal vollständig am Modell angewandt wurden, hat der Nutzer die Möglichkeit, die Ergebnisse dieser Rechnung, also die kalkulierte und optisch sichtbar gemachte Wirkung der Randbedingungen am FE-Modell auszuwerten. Danach steht die Optimierung der Form des FE-Modells zur besseren Anpassung an die gegebenen Randbedingungen, anhand der im Postprozessor visualisierten ersten Ergebnisse, bevor.

Das erneute Präprozessieren ist technisch gesehen der wichtigste Arbeitsschritt. In ihm zeigt sich die Qualität des gesamten Algorithmus. Es geht darum, im FE-Programm zu versuchen, möglichst genau den Vorgang nachzumachen, der in vivo am Knochen passiert. Mechanisch stark beanspruchte Elemente können dabei z.B. im Modell verstärkt werden und weniger stark beanspruchte Elemente können aus dem Modell gelöscht werden, bis sich nach und nach eine Form ergibt, die den gegebenen Randbedingungen optimal standhält (Schumacher 2013). Diese nachträglichen ergebnisorientierten Veränderungen an den Elementen, die im FE-Modell vorgenommen werden müssen, können grundsätzlich jedoch nur in einem erneuten Akt des Präprozessierens vorgenommen werden. Dieses unterscheidet sich vom ersten Präprozessieren darin, dass im ersten Präprozessieren nur das FE-Modell erstellt und Randbedingungen angebracht wurden. Im erneuten Präprozessieren hingegen werden, aufgrund der Ergebnisse des unmittelbar vorherigen Postprozessors, signifikante technische Veränderungen an der Form oder am Elastizitätsmodul des Modells vorgenommen und es werden anschließend wieder die bekannten Randbedingungen am Modell angebracht. Der im Anhang dieser Arbeit zu findende Algorithmus 2 (Tabelle 5), der mit einem festen Elastizitätsmodul arbeitet und selektiv randomisiert Elemente deaktiviert, arbeitet zum erneuten Präprozessieren beispielsweise mit dem folgenden Code (siehe Tabelle 5 – Zeile 338 bis 394):

Tabelle 3: Auszug aus Algorithmus 2 (Tabelle 5)

Zeile	Befehl
338	/solu
339	ekill,all
340	esel,s,live
341	nsel,all
342	lsclear,all
343	LSSOLVE,1,n,1
344	/POST1
345	*do,ls,1,n
346	LCDEF,ls,ls
347	*enddo
348	*do,ls,1,n
349	LCASE,ls
350	*enddo
351	SUMTYPE, PRIN
352	*do,ls,1,n
353	LCOPER,MIN,ls
354	*enddo
355	/title,Etappe %etappe%, Iteration Nr. %iter%, Kill
356	!*
357	PLNSOL,S,3,0,1.0
358	!*
359	
360	!*
361	/CONT,1,50,stress_minimum_%etappe%,0,0
362	/REPLOT
363	!*
364	
365	nsel,all
366	etable,tabelle,s,3
367	ESEL,S,ETAB,TABELLE,stress_minimum_%etappe%*1000,s_2_%etappe%, ,0
368	*GET,e_zahl,elem,0,count
369	*if,e_zahl,gt,0,then
370	nsle,s,all
371	esln,s,0,all
372	cm,alive,elem
373	/solu
374	ealive,all
375	esel,s,live
376	nsel,all
377	lsclear,all
378	LSSOLVE,1,n,1

```
379    /POST1
380    *do,ls,1,n
381    LCDEF,ls,ls
382    *enddo
383    *do,ls,1,n
384    LCASE,ls
385    *enddo
386    SUMTYPE, PRIN
387    *do,ls,1,n
388    LCOPER,MIN,ls
389    *enddo
390    /title,Etappe %etappe%, Iteration Nr. %iter%, Alive
391    !*
392    PLNSOL,S,3,0,1.0
393    !*
394    *endif
```

In diesem Code werden zunächst aus einem vom Nutzer vorbestimmten Intervall zufällig Elemente deaktiviert (Zeile 339). Danach werden alle Lastschritte am selben Modell durchgerechnet (Zeile 340-354). Anschließend wird das Ergebnis graphisch dargestellt (Zeile 355-363). Daraufhin werden deaktivierte Elemente, die an zu sehr belasteten Elementen grenzen, wieder reaktiviert (Zeile 365-374) und das Modell wird noch ein Mal mit allen Lastschritten durchgerechnet (Zeile 375-389). Dann wird das Ergebnis wieder visualisiert (Zeile 390-394).

Nach einem erneuten Präprozessieren macht es wieder Sinn, das Modell, aufgrund der Veränderungen, die man an ihm vorgenommen hat, noch einmal zu simulieren und auszuwerten. Wie man sieht, ist es also zweckdienlich, mehrmals den gesamten Ablauf des Präprozessors, Gleichungslösers und des Postprozessors zu wiederholen, um sich dadurch schrittweise in der Form einem Optimum zu nähern. Diese immer wiederkehrende Formoptimierung und Kalkulation ist die wichtigste Aufgabe eines Algorithmus, denn diese vielen Schritte sollen möglichst autonom ablaufen.

Jeder Durchlauf mit je einem Präprozessor, einem Gleichungslöser und einem Postprozessor wird insgesamt eine „Iteration“ genannt. Das heißt also, dass um einer optimalen Form näher zu kommen, mehrere Iterationen benötigt werden, in denen jedes Mal

1. in einem erneuten Präprozessieren, aufgrund der Ergebnisse eines vorangegangenen Postprozessors, Veränderungen am Modell – also Maßnahmen der Formoptimierung des Modells – vorgenommen werden,
2. mit dieser veränderten Form des Modells unter den im ersten Präprozessor definierten Randbedingungen nochmal gerechnet wird (Gleichungslöser) und
3. die Ergebnisse des Gleichungslösers mit der veränderten Form noch einmal angezeigt werden, um Rückschlüsse für das nächste Präprozessieren ziehen zu können (siehe Abbildung 12).

Dieses Wiederholen von Iterationen kann sowohl vom Benutzer selber vollführt werden, oder man nutzt die Möglichkeit, die in vielen, wenn nicht in allen FE-Programmen, gegeben ist,

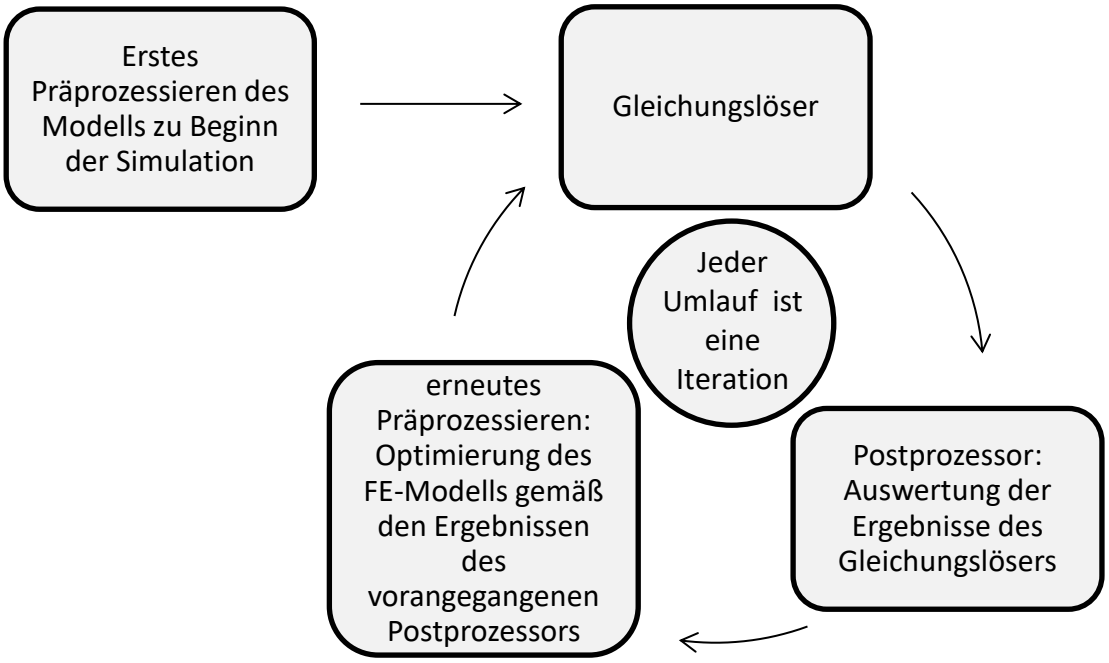


Abbildung 12: Arbeitsablauf einer Simulation mit mehreren Iterationen

einen Algorithmus zu schreiben, der dafür sorgt, dass diese Iterationen automatisch wiederholt werden.

3.2 Algorithmen

Algorithmen sind Informationen in einer für einen Computer verständlichen Sprache, die in einem Programm, angewandt werden können, um eine bestimmte Abfolge abstrakter Befehle

an dieses Programm automatisch zu erteilen und damit ein Problem zu lösen (Pomberger 2008). In unserem konkreten Fall heißt das, dass ein Algorithmus, automatisch, und vorausgesetzt der Nutzer hat das erste Präprozessieren selber vorgenommen, die notwendigen Befehle für den Gleichungslöser, Postprozessor und das erneute Präprozessieren an das FE-Programm erteilt, und zwar in so vielen Iterationen, wie der Nutzer es wünscht.

3.2.1 Die Notwendigkeit eines Algorithmus

Theoretisch können sämtliche Befehle, die der Nutzer an das FE-Programm erteilen möchte, um sein Modell zu simulieren, auch manuell, also vom Nutzer selber erteilt werden. Die Vorteile eines Algorithmus in Bezug auf die Fragestellung dieser Arbeit liegen aber vor allem bei der Zeitersparnis bei den Simulationen, gerade dann, wenn es darum geht in vielen Iterationen zu rechnen, wie es beim Thema dieser Arbeit der Fall ist.

Wenn man voraussetzt, dass die Knochenapposition- und Resorption durch (unter anderem) mechanische Faktoren determiniert ist, so lässt sich im Umkehrschluss auch sagen, dass diese Apposition und Resorption letztlich durch einen Algorithmus berechnet werden kann, weil auch angenommen werden muss, dass die Theorien und Gesetze, die diese Apposition und Resorption beschreiben, gleich bleiben. Wenn diese Theorien und Gesetze sich letztlich als mechanische Formeln beschreiben lassen, so wird mit einem Algorithmus, der genau diese mechanischen Formeln wiedergibt, letztlich auch genau der Vorgang beschrieben, der in vivo im Knochen stattfindet. Aufgrund des aktuellen Standes der FE-Softwaretechnik muss an dieser Stelle, trotz der Erkenntnis, dass mechanische Effekte nicht alleinig Einfluss auf Knochenumbauten haben, eben von dieser vereinfachten Hypothese ausgegangen werden.

Technisch gesehen sind Algorithmen auch deshalb notwendig, weil Programmiersprachen von FE-Programmen häufig über sehr viele einzelne Befehle funktionieren, die nur sehr schwer zusammengefasst werden können. Der Nachteil von solchen Algorithmen mit ihren sehr vielen kleinen Befehlen ist jedoch, dass diese allgemein schnell sehr lang und unübersichtlich werden können. Gegebenenfalls bietet es sich an, Algorithmen deshalb zu kommentieren.

3.2.2 Anforderungen an einen Algorithmus

Generell muss bei der Entwicklung jedes Algorithmus Wert auf eine sogenannte „strukturierte Programmierung“ gelegt werden (Dahl et al. 1972). Das bedeutet letztendlich, dass in der Entwicklung eines Algorithmus, die Fehleranfälligkeit verringert und die Übersichtlichkeit und die Wartungsfreundlichkeit gesteigert werden muss. Außerdem legte Pomberger sechs Anforderungen fest, die an Algorithmen aller Art gerichtet sind: Korrektheit, Vollständigkeit, Eindeutigkeit, Ausführbarkeit, statische/dynamische Endlichkeit und Effizienz (Pomberger 2008). Neben der Größe und der Dimensionszahl zu berechnender FE-Modelle, hat nämlich auch die Schreibweise des Algorithmus einen sehr großen Einfluss auf den Rechenaufwand und die Rechendauer. Bei der Entwicklung gilt es demnach, wie auch beim Schreiben von allen anderen Programmen in der Informationstechnologie, dass darauf zu achten ist, den Algorithmus so kurz wie möglich zu halten und alle Befehle, die nicht notwendig sind oder keinen Nutzen für die Simulation haben, zu entfernen.

Insbesondere sollte in einem speziell für Knochensimulationen zu entwickelnden Algorithmus darauf geachtet werden, dass dieser

- möglichst autark rechnet und den Nutzer nur unabwendbar notwendige Parameter abfragt,
- auf alle Patientenfälle ohne Änderung gleichermaßen anwendbar ist und
- möglichst exakte Simulationsergebnisse berechnet.

4. Ergebnisse

4.1 Funktionen in Algorithmen für das erneute Präprozessieren

Die wichtigste Diskussion und die eigentliche Schwierigkeit bei der Entwicklung eines Algorithmus zum Simulieren von Knochenveränderungen in vivo ist der Aufbau des Algorithmus, in Bezug auf das erneute Präprozessieren (Schumacher 2013). Denn das erste Präprozessieren ist das Darstellen des klinischen Modells als FE-Modell mit Angabe aller Randbedingungen; der darauffolgende Gleichungslöser und der Postprozessor stellen lediglich die Berechnung und Anzeige der Ergebnisse des im ersten Präprozessieren angegebenen FE-Modells dar. Die eigentliche Frage ist also, was im erneuten Präprozessieren konkret am Modell gemacht werden kann und soll, um die Form des Modells gemäß den gegebenen Randbedingungen zu optimieren. Um die gegebenen Möglichkeiten des erneuten Präprozessierens ausloten zu können, lässt es sich nicht vermeiden, diese Möglichkeiten anhand eines konkreten FE-Programms aufzuzeigen. Im Fall dieser Arbeit wurde mit dem FE-Programm ANSYS 14.0 gearbeitet, welches mit der Programmiersprache APDL arbeitet. Die gezeigten Techniken sind aber auch genauso oder in ähnlicher Form in anderen FE-Programmen zu finden.

Zur Optimierung von FE-Modellen gibt es in ANSYS verschiedene Methoden, die alle in einem Algorithmus verwendet werden können. Prinzipiell gilt jedoch, dass vor jeder Operation die Elemente und Knoten, mit denen etwas geschehen soll, selektiert werden müssen, unabhängig davon, was mit ihnen gemacht werden soll. Das funktioniert bei Knoten über den Befehl NSEL und bei Elementen über ESEL. Selektionen können generell anhand sehr vieler verschiedener Kriterien erfolgen. Am einfachsten ist das graphische Aussuchen von Elementen und Knoten per Mausklick durch den Nutzer. In einem automatisierten Algorithmus macht diese Methode jedoch wenig Sinn. Eine andere automatische Art der Selektion hingegen ist die Selektion nach physikalischen Größen, wie z.B. der mechanischen Spannung. Das ist die sinnvollste Art, für Knochensimulationen Knoten oder Elementen zu selektieren und funktioniert so, dass eine Unter- und eine Obergrenze für die Spannung festgelegt werden kann und dann alle Elemente oder Knoten, die innerhalb dieser Grenzen liegen, für das weitere Geschehen selektiert sind. Nachfolgend werden nun die einzelnen zur Verfügung stehenden Methoden zur Optimierung von Modellen vorgestellt.

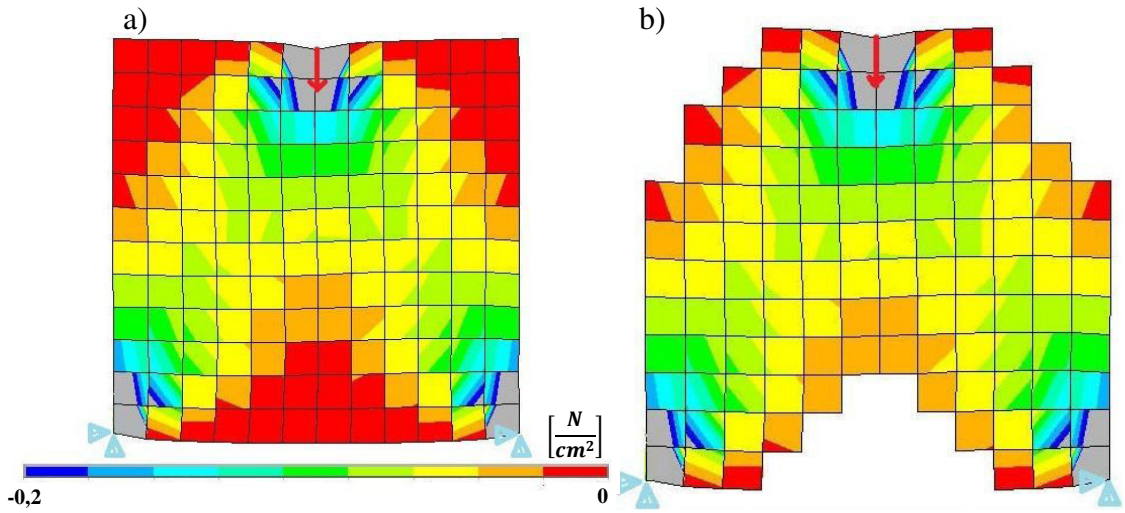


Abbildung 13: ANSYS – roter Pfeil = Kraft (1 N), blaue Keile = Lager in x- und y-Richtung, Modell-Größe 10x10cm; Drucksimulation a) Modell vor dem Deaktivieren ungebrauchter Elemente b) im Vergleich zum Modell nach dem Deaktivieren. In diesem Fall wurden alle Elemente aus einem bestimmten Spannungsbereich, in dem der Druck sehr gering ist (rotes Ende der Skala) automatisch selektiert und deaktiviert.

4.1.1 Deaktivieren und Reaktivieren von Elementen

Das Deaktivieren von Elementen ist eine der geläufigsten Methoden. Dabei werden einzelne Elemente aus dem FE-Modell herausgenommen. Dies funktioniert über den Befehl EKILL. Graphisch betrachtet fehlen dann diese Elemente, doch technisch gesehen sind sie immer noch da und haben nur einen um den Faktor 10^6 verringerten Elastizitätsmodul (siehe Abbildung 13). Durch das Deaktivieren werden Elemente nicht gänzlich entfernt, da dann das FE-Modell nicht mehr berechnet werden könnte. Nach dem Deaktivieren eines Elementes erhöht sich natürlicherweise die Spannung in den umliegenden Elementen rund um das deaktivierte Element, da dieselbe Last nun über weniger Elemente verteilt ist (Shokrieh 2014).

Das Deaktivieren als Funktion in einem Algorithmus bietet sich z.B. an Stellen an, in denen von einer sehr geringen Last ausgegangen wird, theoretisch wird auch angenommen, dass Knochen dort wächst, oder vorhanden ist, wo auch tatsächlich eine Last zu tragen ist. Andernfalls müssten Knochenareale, die nicht oder zu wenig belastet sind, atrophieren.

Sollte man deaktivierte Elemente reaktivieren wollen, da die Spannung in den umliegenden Elementen zu sehr angestiegen ist, muss man den Befehl EALIVE nutzen. Prinzipiell können Elemente beliebig oft deaktiviert und reaktiviert werden. Nach dem Reaktivieren sind sie wieder in der Form und mit den physikalischen Eigenschaft vorhanden, wie sie vor dem Deaktivieren waren, vorausgesetzt das Netz des FE-Modells blieb unverändert. An Stellen, wo nie Elemente existiert haben, können auch keine neuen Elemente geschaffen werden.

4.1.2 Zufälliges Deaktivieren von Elementen

Eine in den Ergebnissen interessante, aber bisweilen wissenschaftlich gesehen fragliche Idee zur Modelloptimierung ist es, Elemente, anstatt sie nach strengen Spannungsgrenzen zu selektieren und zu deaktivieren, diese Spannungsgrenzen im Laufe von Iterationen zu ändern, und sogar innerhalb bestimmter Intervalle von vordefinierten Spannungsgrenzen Elemente per Zufall zu deaktivieren. Diese Methode ist in Anbetracht der Fragestellung dieser Arbeit, aber auch generell, skeptisch zu betrachten, da durch das zufällige Selektieren von Elementen das Grundkonzept der kausalen, durch bestimmte Spannungen bedingten Veränderungen in knöchernen Strukturen, infrage gestellt wird. Andererseits führten in dieser Arbeit jedoch Versuche mit dieser Methode zu Ergebnissen, in denen sich spongiosa- und kompaktaähnliche Strukturen abzeichneten (siehe Abbildung 14). Der Algorithmus 2, mit dem diese Formen konstruiert wurden, ist im Anhang zu finden (siehe Tabelle 5).

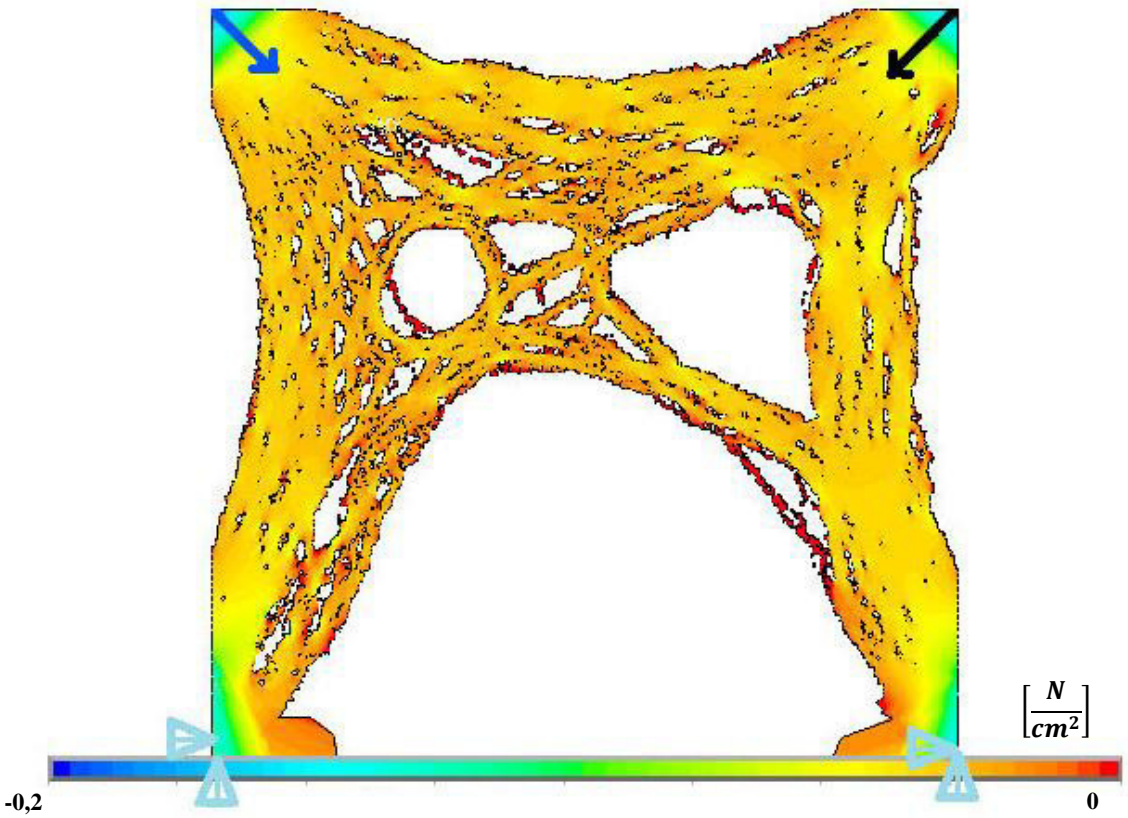


Abbildung 14: ANSYS – blauer Pfeil/schwarzer Pfeil = Kraft (je 1 N), blaue Keile = Lager in x- und y-Richtung, Modell-Größe 10x10cm; Drucksimulation in zwei Lastschritten (jede Kraft ist ein eigener Lastschritt); gemäß dem Algorithmus 2 wurden selektiv randomisiert Elemente deaktiviert; als Ergebnis zeigt sich mittig des Modells ein spongiöses Muster, welches nach außen hin immer kompakter wird

4.1.3 Änderung des Elastizitätsmoduls

Eine andere ebenso interessante Theorie, ist es, statt Elemente immer vollständig zu deaktivieren oder zu reaktivieren, den Elastizitätsmodul von Elementen zu variieren. Im Hinblick darauf, dass durch das Deaktivieren der Elastizitätsmodul eines Elements nur noch ein millionstel des Ursprungswertes beträgt, ist es sicher wichtig zu wissen, wie die Ergebnisse wären, würde man den Elastizitätsmodul von Elementen z.B. in mechanisch wenig beanspruchten Arealen, um eine gewisse Prozentzahl (20%, 50% oder 80%) senken, beispielsweise um weniger dichte Knochenregionen zu simulieren. In dieser Arbeit wurde deswegen auch ein Algorithmus entwickelt, der zunächst versucht den Elastizitätsmodul der Elemente zu ändern (Algorithmus 1, Tabelle 4). Sollte dieser Schritt die Modellform nicht genügend optimieren, werden dann auch Elemente deaktiviert.

Technisch gesehen besteht die Möglichkeit dazu durch den Befehl MPCHG. Allerdings steht, vor Nutzung dieser Option zunächst die Entscheidung an, welche Ergebnisse man für die Knochensimulationen erwartet. Nutzt man die Option zur Änderung des Elastizitätsmoduls für wenig belastete Elemente, so wird der Algorithmus in wenig belasteten Regionen des Modells weniger Elemente deaktivieren und häufiger die Elastizitätsmodule besagter Elemente reduzieren. Bleibt man jedoch bei der konventionellen Technik der Formoptimierung mit einheitlichem Elastizitätsmodul für alle Elemente, so ist damit zu rechnen, dass der Algorithmus häufiger Elemente deaktiviert und damit möglicherweise ein gänzlich neues Muster des Modells schafft. Tendenziell wird bei Simulationen, in denen eine Änderung des Elastizitätsmoduls vorgenommen wird, ein weniger feines Netz benötigt, als in Simulationen mit einheitlichem Elastizitätsmodul, weil in letzteren Simulationen sämtliche spongiöse Mikroarchitektur nur aufgrund eines feinen Netzes konstruiert werden kann.

Prinzipiell sind beide Techniken in Algorithmen möglich. Die Entscheidung, welche der beiden Techniken gewählt wird, obliegt dem Nutzer. Sie sollte davon abhängig gemacht werden, wozu das fertig simulierte Modell verwendet werden soll. Falls eine Nutzung im Sinne des CAM-Verfahrens (Computer Associated Manufacturing) angestrebt wird, kann es hinderlich sein, den Algorithmus eine Spongiosa simulieren zu lassen, die technisch nicht hergestellt werden könnte. Soll die Simulation jedoch lediglich zur visuellen Veranschaulichung und Vorhersage von zukünftigen Kochenveränderungen dienen, so kann es Sinn machen, einen einheitlichen Elastizitätsmodul zu nutzen.

4.1.4 Lastschritte

Eine der wichtigsten und zugleich kompliziertesten Optionen bei FE-Simulationen allgemein und bei der Fragestellung dieser Arbeit im Speziellen sind sogenannte „Lastschritte“. Generell kann man sagen, dass Lastschritte Randbedingungen zu verschiedenen Zeiten darstellen, wobei mit dem Wort „Zeit“ nicht Chronologie gemeint ist, sondern eher verschiedene Situationen unabhängig davon, wann sie auftreten (Tadeusz Stolarski 2006). Da es sich hierbei um ein abstraktes Phänomen handelt, welches häufig zu Missverständnissen führt, soll ein einfaches Beispiel verdeutlichen, was gemeint ist:

Mit einem FE-Programm soll die Neukonstruktion einer Autobahnbrücke, die zwei Berge über einem Tal miteinander verbinden soll, überprüft werden. Ähnlich wie bei der

Fragestellung dieser Arbeit würde man entsprechend ein Ausgangsmodell definieren und danach sämtliche Kräfte und Lager, die auf diese Brücke wirken, im Präprozessor am FE-Modell anbringen.

Das sind zunächst einmal die Kräfte der Fahrzeuge, die über die Brücke fahren. Sie wirken gemäß der Erdanziehung senkrecht nach unten. Doch wenn man z.B. annimmt, dass es zu einem anderen Zeitpunkt einen verkehrsfreien Tag geben kann, an dem keine Fahrzeuge fahren, dafür aber starker Seitenwind gegen die Konstruktion drückt, ändert das die Randbedingungen der Konstruktion grundlegend. Denn in dieser Situation gibt es einzig Kräfte, die horizontal von der Seite wirken, aber, abgesehen vom Eigengewicht der Brücke, keine Kräfte, die vertikal nach unten wirken. Wie man sieht, gibt es also verschiedene Situationen, in denen die Brücke ganz unterschiedlichen Kräften ausgesetzt ist. Doch letztlich muss dieselbe Brücke so konstruiert werden, dass sie für alle Situationen ausgelegt ist und immer standhält.

In diesem Beispiel stellen die Randbedingungen, die in der Situation, in der die Fahrzeuge über die Brücke fahren und kein Seitenwind herrscht, insgesamt für sich genau einen Lastschritt dar. Der zweite Lastschritt in diesem Beispiel ist die Situation, in der die Brücke unbefahren ist und nur Seitenwind herrscht.

Dieses Beispiel ist auf Knochensimulationen ebenso übertragbar. Bezogen auf einen Unterkieferknochen hieße das z.B., dass die Ruheschwebelage ein Lastschritt ist, das Sprechen ein zweiter und das Kauen ein dritter. Möglicherweise müsste man sogar, was das Kauen angeht, um genau zu sein, das Mahlen mit den Seitenzähnen als einen eigenen Lastschritt definieren und das Abbeißen von Speisen mit den Frontzähnen als einen weiteren.

Prinzipiell müssen Lastschritte immer nacheinander, jeweils isoliert für sich, im Gleichungslöser am FE-Modell berechnet werden. Es gibt dabei je einen eigenen Gleichungslöser für jeden Lastschritt. Die Ergebnisse der einzelnen Gleichungslöser werden zusammengerechnet und im Postprozessor insgesamt am selben Modell angezeigt. Der Postprozessor nach den einzelnen Gleichungslösern stellt also immer das Gesamtergebnis in Bezug auf alle Lastschritte dar. Der Ansatz, zu versuchen, alle Randbedingungen aller Lastschritte auf einmal an einem FE-Modell anzubringen, und dann das Modell zu simulieren, mit der Absicht, auf diese Weise das Modell auf alle Situationen zu prüfen, ist falsch und führt zu einem anderen Ergebnis, als wenn die Lastschritte einzeln berechnet werden (siehe Abbildung 15). Es müssen alle Lastschritte einzeln berechnet werden und die Addition der

Ergebnisse, also z.B. der Spannungen von jedem einzelnen Lastschritt an jedem einzelnen Element ergibt im Postprozessor ein Bild, das die Gesamtbelastung aller Lastschritte auf ein und dasselbe Modell darstellt (Groth 2002).

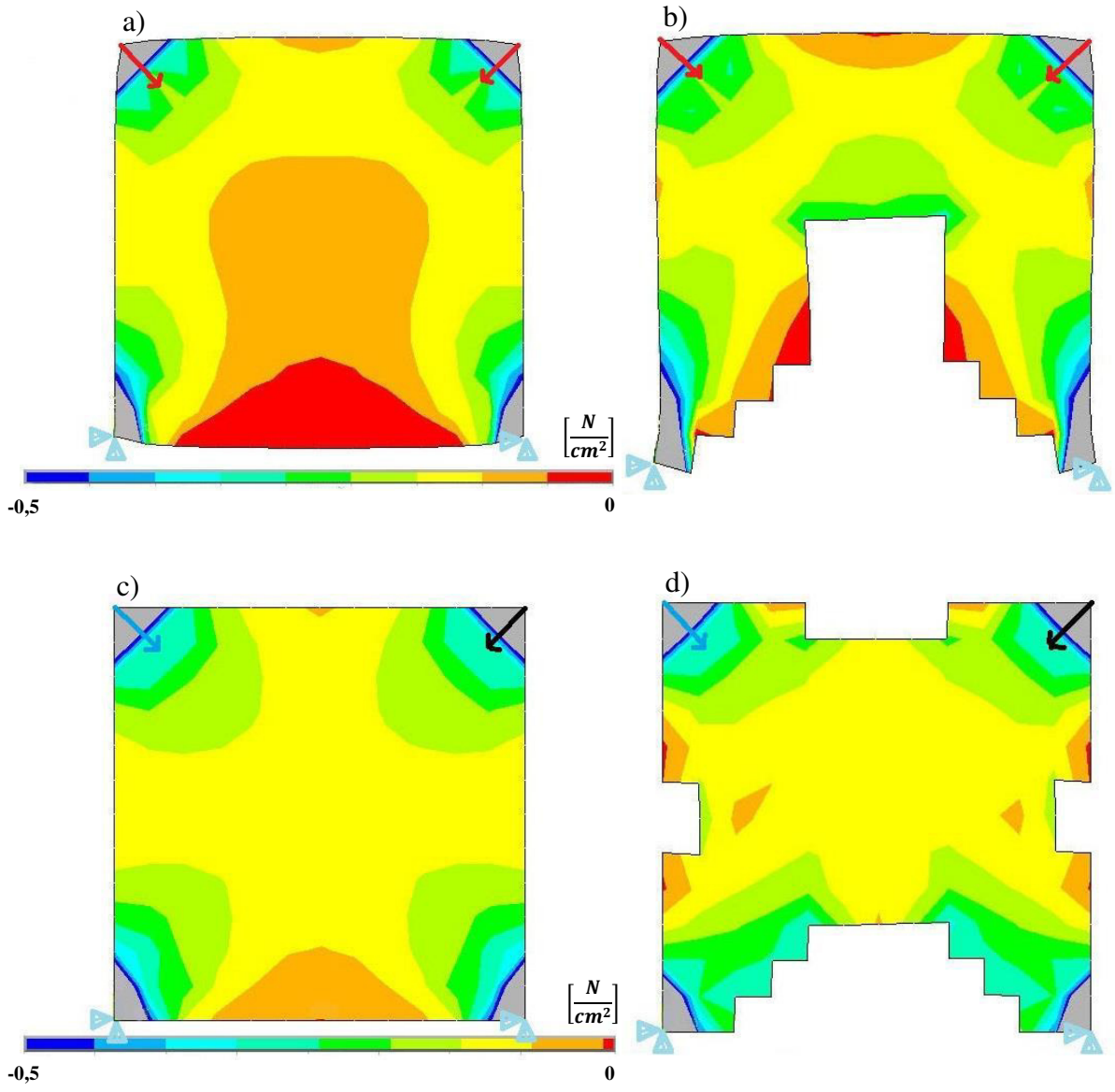


Abbildung 15: ANSYS – rote Pfeile/blauer Pfeil/schwarzer Pfeil = Kräfte (je 1 N), blaue Keile = Lager in x- und y-Richtung, Modell-Größe 10x10cm; Drucksimulation; Abbildung a) und c) zeigen dasselbe FE-Modell. Abbildung d) und b) zeigen jeweils den ersten Optimierungsschritt nach dem Gleichungslöser in Bezug auf das jeweils linke Modell. Man sieht signifikante Unterschiede in den Ergebnissen, die dadurch begründet sind, dass in Abbildung a) die beiden Kräfte insgesamt als ein einziger Lastschritt gerechnet wurden und sie in Abbildung c) jeder für sich ein eigener Lastschritt sind.

Lastschritte werden generell im ersten Präprozessor definiert und gespeichert. Im Gleichungslöser werden automatisch, sofern man die Lastschritte im ersten Präprozessor angegeben hat, alle Lastschritte einzeln und nacheinander berechnet und die Ergebnisse jeweils gespeichert und beim Postprozessor miteinander addiert. Das Vorgehen im erneuten Präprozessieren ist das gleiche, wie auch bei Simulationen ohne Lastschritte bzw. mit nur einem Lastschritt.

Die technische Option der Lastschritte in Algorithmen spielt besonders im Hinblick auf klinische Fragestellungen eine große Rolle, da auch dort häufig, wenn nicht immer, in den Modellen mehrere Situationen im Sinne von mehreren Lastschritten denkbar sind.

4.1.5 Netzverfeinerung

Eine ebenfalls wichtige Technik, die in einem Algorithmus ihre Verwendung finden kann, ist das Zerteilen einzelner Elemente. Wenn man in der ersten Iteration einer Simulation von relativ großen Elementen ausgeht, kann man nicht verhindern, dass nach mehreren Iterationen und einigen deaktivierten Elementen, das Modell Kanten aufweist, in denen die Elemente Inhomogenitäten in der Spannung aufweisen. Um dieser Schwierigkeit zu begegnen, können diese Elemente um ein vom Nutzer festgelegtes Maß zerteilt werden (Fröhlich 2005). Diese Option funktioniert über den Befehl EREF. Terminologisch nennt man ein Element, bevor es zerkleinert wird, also solange es noch ein Ganzes ist, „Mutter-Element“ und die kleineren Elemente, die nach der Verfeinerung insgesamt den Platz des Mutter-Elements einnehmen, „Töchter-Elemente“ (siehe Abbildung 16). Kleine Verzerrungen an Elementen, für die eigentlich keine Netzverfeinerung vorgesehen war, sind nicht ausgeschlossen. Töchter-Elemente erben immer jeweils alle Eigenschaften, die auch das Mutter-Element hatte, wie z.B. Knotenanzahl und Elastizitätsmodul. Es lässt sich bestimmen, ob die Elemente nach ihrer Verfeinerung die gleiche Form haben sollen, wie vorher, oder ob das FE-Programm automatisch Änderungen im Sinne der Optimierung des Netzes vornehmen darf. Beispielsweise macht es manchmal Sinn, bei einem 2-D-Modell, das mit viereckigen Elementen vernetzt ist, dem Programm zu erlauben, an bestimmten Stellen nach einer Netzverfeinerung teilweise dreieckige Töchter-Elemente einzusetzen, damit der Modellrand einen sanfter Verlauf haben kann. Grundsätzlich können Töchter-Elemente nachträglich nicht

mehr zum ursprünglichen Mutter-Element zusammengefügt werden. Es gibt also keine Option auf eine „Vergröberung“ des Netzes.

Prinzipiell bedeutet eine Netzverfeinerung immer eine Steigerung der Anzahl der Elemente (Fröhlich 2005). Das wiederum bedeutet, dass im Gleichungslöser häufiger gerechnet werden muss, also dauert die gesamte Rechnung länger. Dieser Aspekt lässt sich sowohl bei 2-D-

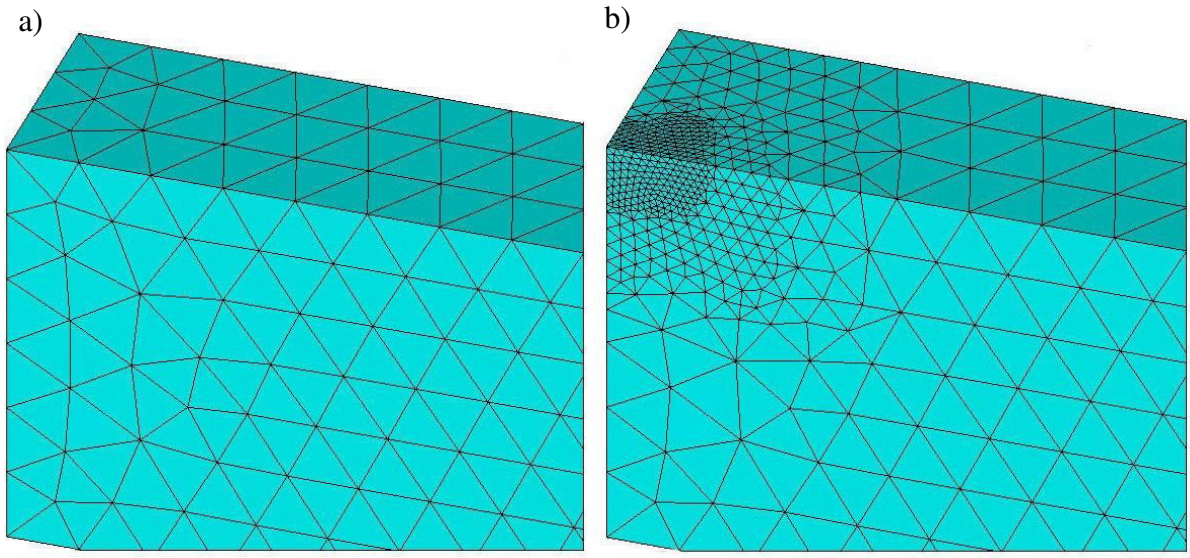


Abbildung 16: ANSYS – FE-Modell mit a) gleich großen Tetraeder-Elementen b) dasselbe Modell nach lokaler einmaliger, zweimaliger und dreimaliger Netzverfeinerung

Modellen, als auch bei 3-D-Modellen beobachten, ist jedoch bei 3-D-Modellen deutlich stärker ausgeprägt. Unter Umständen kann das dazu führen, dass ein handelsüblicher Rechner für die Berechnung einer einzigen Iteration eines 3-D-Modells mehrere Stunden, bis hin zu mehreren Tagen benötigt. Deswegen ist, je nach Größe des Modells und Anzahl der Iterationen, die gerechnet werden sollen, gegebenenfalls der Einsatz von sogenannten „Supercomputern“ zu erwägen.

Allerdings sollte man wissen, dass auch die FE-Programme immer eine feste Grenze hinsichtlich der Anzahl der zu berechnenden Knoten haben, denn jede Verfeinerung bedeutet ja immer, nicht nur eine Zunahme der Elemente, sondern auch eine Zunahme der Knoten. Ist diese maximale Anzahl der Knoten erreicht, so ist keine weitere Netzverfeinerung mehr möglich und etwaige Befehle zur weiteren Netzverfeinerung werden entweder ignoriert oder mit einer Fehler-Meldung beantwortet.

Es empfiehlt sich deswegen, Rechnungen mit einem relativ groben Netz zu beginnen und die Möglichkeit der Netzverfeinerung nicht immer am gesamten Modell, sondern im Laufe der Iterationen bewusst und nur an ausgewählten Stellen, wo es Sinn macht, einzusetzen. Diese Stellen sind am ehesten dort, wo im Postprozessor Inhomogenitäten der Spannung im Modell zu finden sind (siehe Abbildung 17).

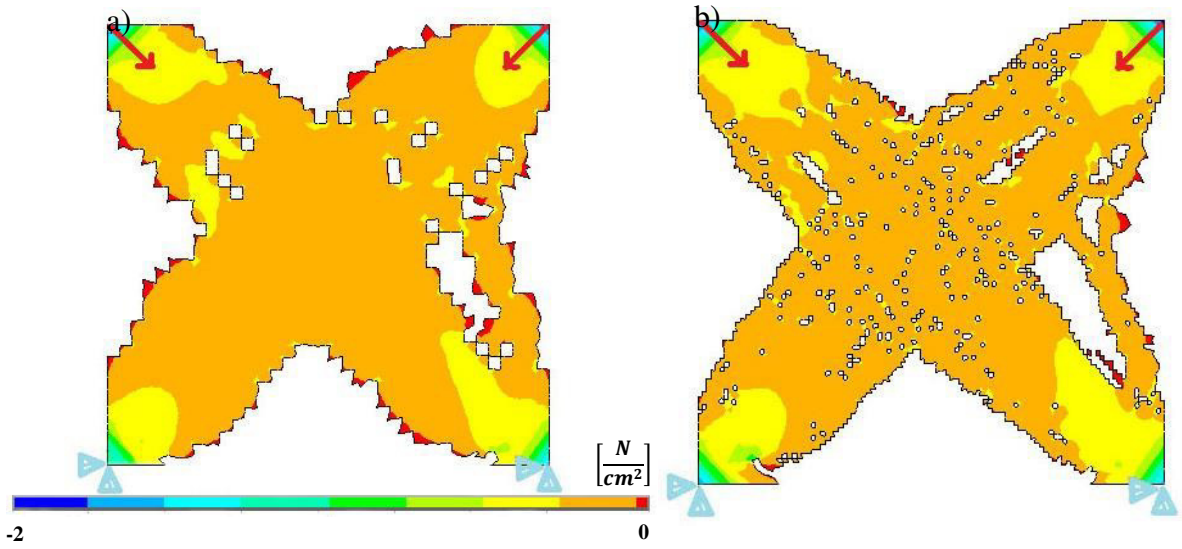


Abbildung 17: ANSYS – rote Pfeile = Kraft (je 1 N), blaue Keile = Lager in x- und y-Richtung, Modell-Größe 10x10cm; Drucksimulation; a) Spannungs-Inhomogenitäten an den Rändern (rote Kanten) b) Besserung durch die Netzverferirung (rechts)

4.1.6 Das Abfragen von Werten

Eine weitere nicht zu unterschätzende Funktion, die in Algorithmen vorkommen wird, ist die, dass der Algorithmus bestimmte Werte den Nutzer abfragt, sodass also der Nutzer dem Programm bestimmte Werte vorgibt. Diese Funktion ist deshalb wichtig, da in der Testphase von Algorithmen häufig noch nicht ganz klar ist, welche Werte bezüglich der einzelnen Funktionen, beispielsweise beim Selektieren von Elementen anhand ihrer Spannungswerte zum Deaktivieren, zum Ändern des Elastizitätsmoduls, zur Netzverfeinerung, oder auch nur zur Angabe der Anzahl der Iterationen, sinnvoll sind. Deswegen ist es praktisch, den Algorithmus so zu schreiben, dass er den Nutzer alle diese Werte fragt, damit dieser schnell auch mit anderen Werten die gleiche Simulation rechnen kann, ohne am Algorithmus deswegen etwas ändern zu müssen.

4.2 Die Entwicklung eines Algorithmus

Wie man am Beispiel von ANSYS sieht, gibt es also prinzipiell alle Bausteine zur Entwicklung eines Algorithmus, die dem Grunde nach in Frage kämen. Erste Algorithmen, die an vereinfachten dreidimensionalen Modellen die Knochenumbauprozesse bei vereinfachten Zahnbewegungen simulieren können, konnten auch schon erfolgreich getestet werden (Schneider 2002). Letztlich ist jedoch die Entwicklung eines Algorithmus, der exakt die Veränderungen an einem realen Knochenstück berechnen soll, eine Arbeit, bei der nicht auf Anhieb gesagt werden kann, wie dieser Algorithmus konstruiert sein müsste. Deshalb ist hier der Begriff „Entwicklung“ im engeren Sinne zu verstehen. Es empfiehlt sich deshalb, in einer ersten Phase, bei dieser Entwicklung anfangs nur mit Knochenstücken zu arbeiten, die sich auch gut als 2-D-Modelle darstellen lassen, wie z.B. dem oberen Drittel des menschlichen Femur in frontaler Schichtung (siehe Abbildung 18). Dabei sollte zunächst versucht werden, ausgehend von einem einfachen Rechteck oder Quadrat, durch Anbringen der Kräfte, die klinisch an diesem Knochen wirken, den Algorithmus so zu schreiben, dass das Programm aus dem anfänglichen Quadrat oder Rechteck, im Laufe der Iterationen eine Form konstruiert, die dem klinisch vorliegenden Femur möglichst genau ähnelt. In diesem Zuge wird sich zeigen, wie gut der in dieser Arbeit entwickelte Algorithmus tatsächlich arbeitet und inwiefern etwaige Feinabstimmungen darin notwendig sind. Wenn dieser Ansatz geglückt ist, sollte man dann erst auf einfache 3-D-Modelle übergehen und dort ebenfalls

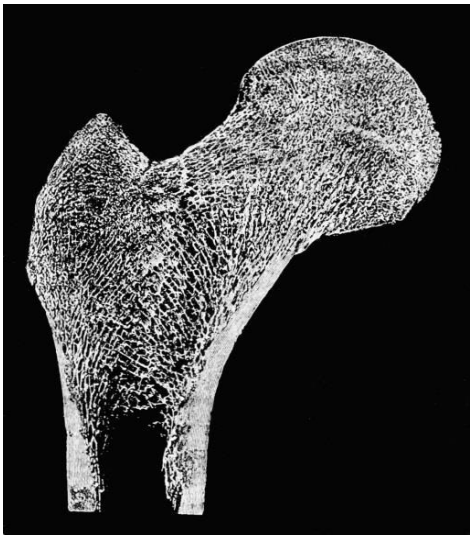


Abbildung 18: Femur-Kopf, Wikimedia Commons, gemeinfrei; man sieht Knochentrabekel, die sich meist an der Richtung der Last orientieren und nach kaudal immer weniger dicht werden.

ausgehend von einfachen Modellen weiter am Algorithmus arbeiten, bis dieser schließlich auch für kompliziertere 3-D-Modelle, wie z.B. dem menschlichen Unterkiefer, genaue Ergebnisse liefert.

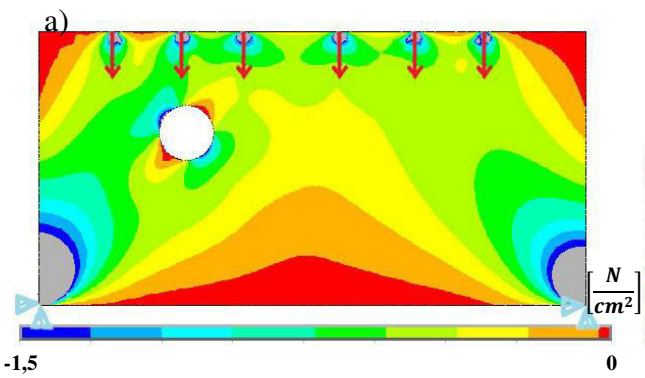
Erst wenn diese erste Phase in Bezug auf 2-D- und 3-D-Modelle vollendet wurde, kann davon ausgegangen werden, dass dieser Algorithmus auch – in einer zweiten Phase – für eine zukünftige klinische Entwicklung genaue Vorhersagen für Knochenmodelle treffen kann.

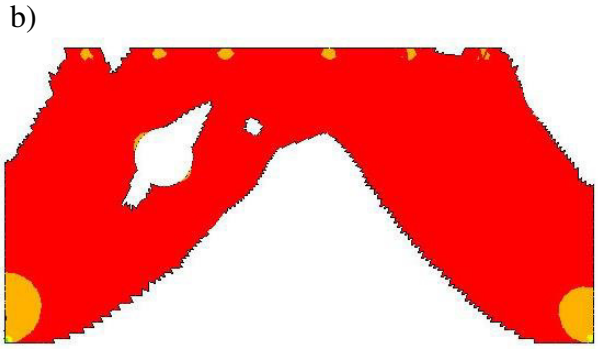
Diese gesamte Arbeit erfordert jedoch sehr viel Zeit und setzt ein biomechanisches Grundverständnis voraus, da die Einschätzung der Beträge und Richtungen der Kräfte eine entscheidende Rolle für die Qualität der Ergebnisse der Simulationen und somit auch für die Entwicklung eines Algorithmus spielt.

4.2.1 Beispiel für einen Algorithmus

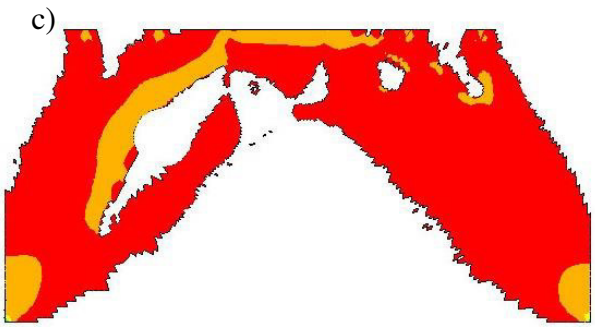
Anhand der Fragestellung einer Straßenbrücke wird in diesem Kapitel gezeigt, wie prinzipiell der Ablauf einer FE-Simulation von der ersten Iteration bis zum gewünschten Ergebnis ist.

Es handelt sich um ein vereinfachtes zweidimensionales Brückenmodell, bei dem davon ausgegangen wird, dass es zwei höher gelegene Areale gibt, die durch ein Tal voneinander getrennt sind, und miteinander verbunden werden sollen. Der für diese Simulation verwendete Algorithmus wurde so konstruiert, dass in mehreren Etappen gerechnet werden kann. Das bedeutet, dass zu Anfang der Simulation der Nutzer dem Programm mitteilen muss, in wie vielen Etappen gerechnet werden soll, und dann noch in jeder Etappe angeben muss, in wie vielen Iterationen gerechnet werden soll. Das Rechnen in Etappen hat den Vorteil, dass sämtliche Parameter, die vom Nutzer bestimmt werden müssen, und mit denen der Algorithmus arbeiten muss, nach jeder Etappe neu bestimmt werden können. Würde man hingegen mit nur einer Etappe arbeiten, liefe die gesamte Simulation für sämtliche Iterationen immer mit denselben Kostanten ab. Abbildung 20 zeigt den schematischen Aufbau des Algorithmus, mit dem die in Abbildung 19 gezeigte Brücke simuliert wurde. Bei diesem Algorithmus handelt es sich um den Algorithmus 2 aus Tabelle 5, mit dem einzigen Unterschied, dass die Elementen nicht randomisiert deaktiviert wurden, sondern nach festen Vorgaben. Ein randomisiertes Deaktivieren von Elementen bot sich hier nicht an, weil in dieser Fragestellung die Erschaffung spongiöser Strukturen keinen Zweck hatte.

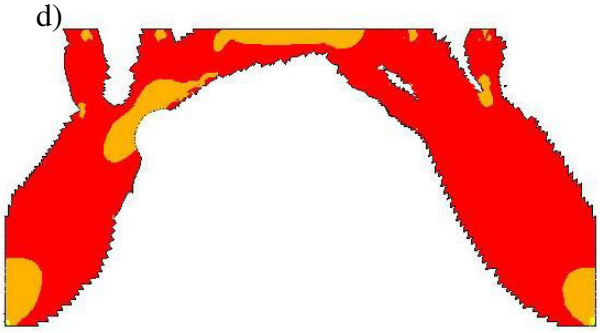




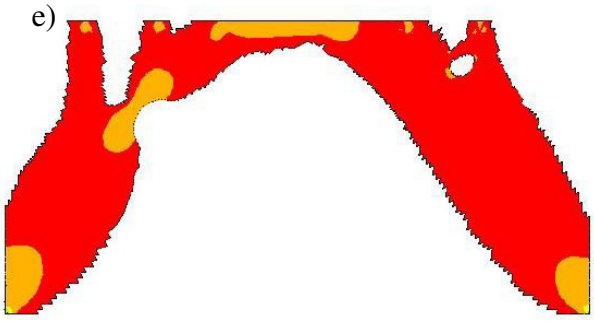
Etappe 1, Iteration Nr. 1



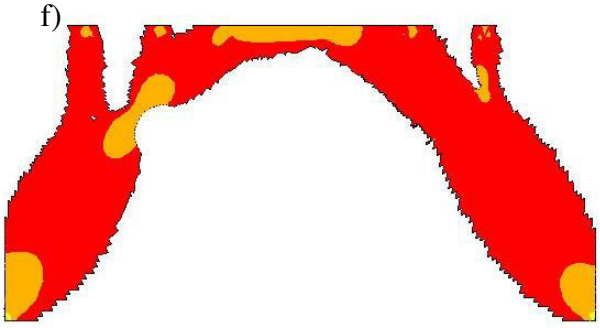
Etappe 4, Iteration Nr. 1



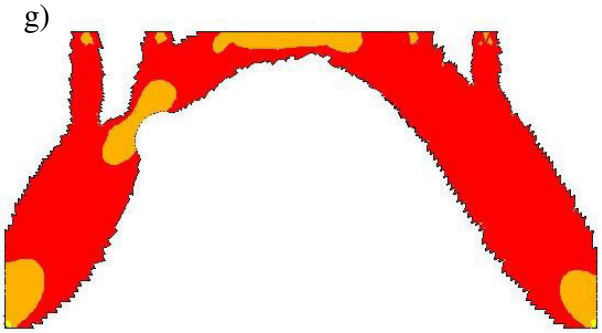
Etappe 4, Iteration Nr. 3



Etappe 7, Iteration Nr. 1



Etappe 7, Iteration Nr. 5



Etappe 13, Iteration Nr. 3

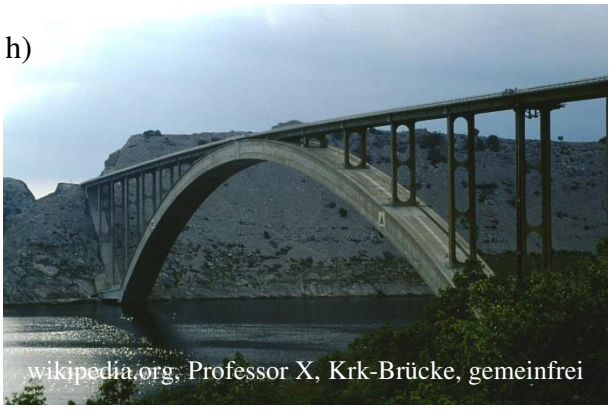


Abbildung 19: ANSYS – rote Pfeile = Kraft (je 1 N), blaue Keile = Lager in x- und y-Richtung, Modell-Größe 20x10cm; Drucksimulation; Entwicklung einer Autobahnbrücke. Das Ergebnis ist vergleichbar mit dem realen Vorbild in Abbildung h)

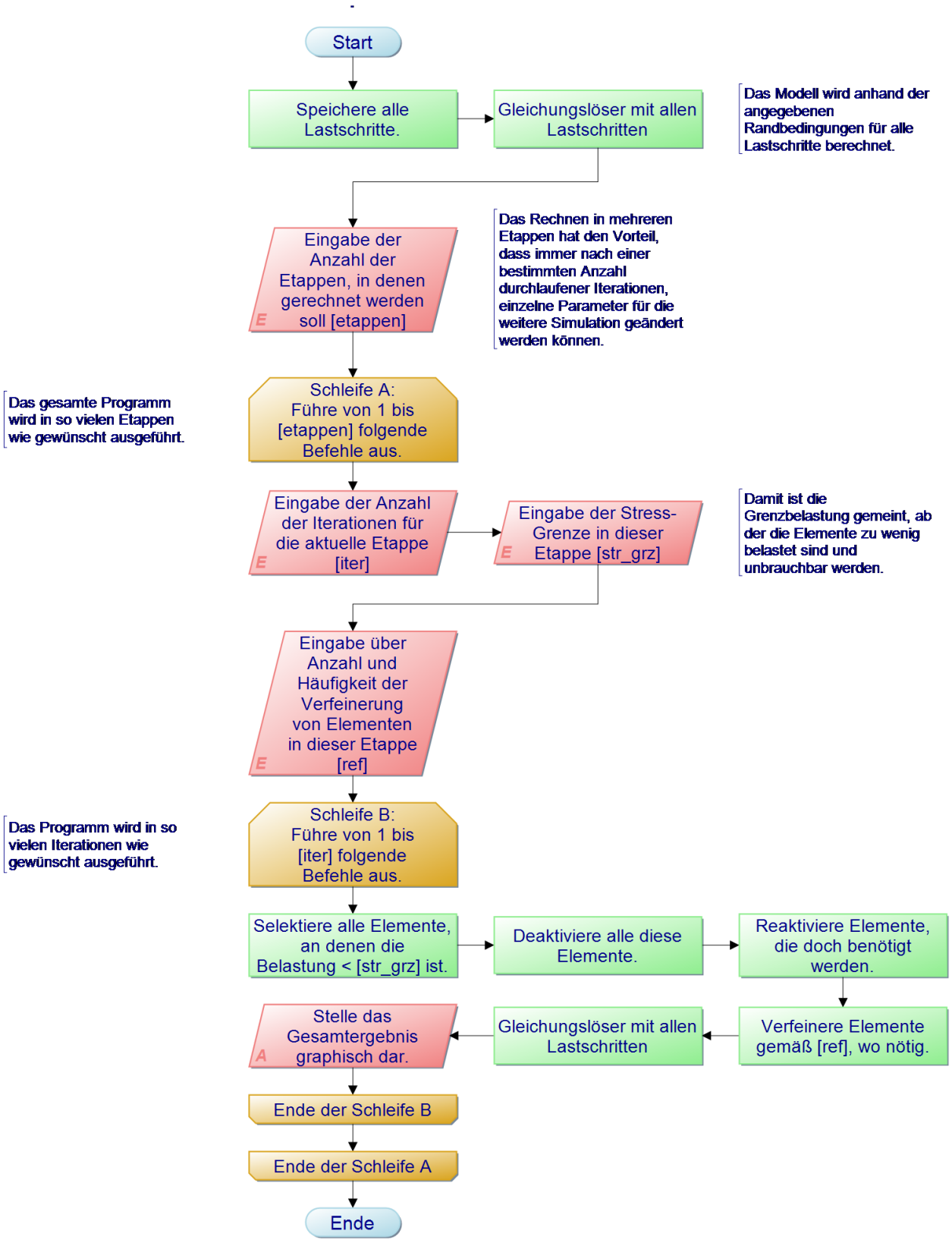


Abbildung 20: Programmablaufplan für den in Abbildung 19 genutzten Algorithmus. Hierbei handelt es sich um eine leichte Abwandlung des Algorithmus 2 aus Tabelle 5.

5. Diskussion und Perspektive

5.1 Diskussion über die verschiedenen Methoden der Übertragung der Randbedingungen in das FE-Programm

Die in Kapitel 2.3.2.1 diskutierte Methode der Elektromyografie funktioniert zwar relativ einfach und ist wenig invasiv. Jedoch ist sie kaum aufschlussreich in Bezug auf auftretende Kraftbeträge. Diese Methode wird daher auch eher zur Befundung der allgemeinen Aktivität eines Muskels verwendet, häufig um im Seitenvergleich zwischen symmetrisch angelegten Muskeln, etwaige Ungleichheiten zwischen zwei Seiten festzustellen. Deswegen kann sie in dieser Form dieser Arbeit nicht dienen.

Eine bislang gut erforschte Methode zur Quantifizierung von Muskelkräften stellt dagegen die Auswertung von Magnetresonanztomographie dar (siehe Kapitel 2.3.2.2). Mit einem entsprechenden Aufwand kann das maximale Kraftpotenzial eines Muskels bestimmt werden. Jedoch nimmt die Auswertung einer Magnetresonanztomographie viel Zeit in Anspruch und es wurde bisher keine Technik entwickelt, diese Auswertungen zu automatisieren.

Die präziseste unter den in dieser Arbeit untersuchten Methoden zur Feststellung der Belastungen an lebendem Knochen stellt die in Kapitel 2.3.2.3 vorgestellte In-vivo-Kräftemessung dar, die am Beispiel von Bergmann diskutiert wurde. Sowohl der Betrag, als auch die Richtung von Kräften, die an einem Knochen wirken, werden in drei Dimensionen gemessen und die Fehlerrate ist gering. Jedoch setzt die Methode die Implantierung der Messinstrumente in den lebenden Organismus voraus. Dies ist, sofern eine Allgemeinanästhesie zur Implantierung notwendig ist, prinzipiell nicht bei allen Patienten möglich, man denke z.B. an Patienten mit schweren Herzrhythmusstörungen. Auch bei Patienten, bei denen medizinisch gesehen eine Implantierung von Messinstrumenten unter Allgemeinanästhesie möglich ist, ist diese Methode, zumindest in der von Bergmann beschriebenen Form, zu invasiv, als dass sie klinisch standardmäßig bei jedem Patienten durchgeführt werden könnte. Allerdings könnte eine technische Weiterentwicklung dieser Methode, in dem Sinne, dass die Invasivität der Implantierung der Messinstrumente gemindert wird und die Messinstrumente kleiner werden, dazu führen, dass auf lange Sicht möglicherweise doch mit dieser Methode gearbeitet werden kann.

Nach Begutachtung der zur Zeit zur Verfügung stehenden Methoden zur Quantifizierung von Muskelkräften kann zusammenfassend gesagt werden, dass es bis jetzt keinen Ansatz gibt, mit dem hinreichend einfach und präzise die klinischen Kraftverhältnisse, die an einem Knochen vorliegen, in ein FE-Programm übertragen werden können. Dieses Problem bietet also weiterhin genügend Potenzial für Forschung.

5.2 Diskussion über die Ergebnisse der Arbeit

Unter Einsatz der in den Kapiteln 4.1.1 bis 4.1.6 erläuterten Techniken ist es in dieser Arbeit gelungen, spezielle für in-vivo-Knochenumbauten funktionierende Algorithmen zu entwickeln. Die entwickelten Algorithmen konnten hauptsächlich nur an zweidimensionalen einfachen Modellen getestet und verifiziert werden (siehe z.B. Kapitel 4.1.2, 4.1.5 und 4.2.1). Es konnte jedoch gezeigt werden, dass grundsätzlich Algorithmen möglich sind, die aus einer neutralen Grundform – bei vorgegebenen Randbedingungen – Objekte simulieren können, die aus biomechanischer Sicht plausibel erscheinen. Die Entwicklung eines Algorithmus, der mit einer hohen Sicherheit realistische zweidimensionale und dreidimensionale Knochenstücke simulieren kann, konnte in dieser Arbeit nicht geleistet werden, weil diese Aufgabe einen unverhältnismäßig größeren Aufwand bedeutet. Es wurden insgesamt zwei Algorithmen entwickelt, die jeder für sich ein anderes Konzept verfolgen. Im Algorithmus 1 kommt nur ein Elastizitätsmodul zum Einsatz und durch die Technik des zufälligen Deaktivierens von Elementen werden spongiosaähnliche Strukturen konstruiert. Im Algorithmus 2 dagegen werden zwar auch Elemente deaktiviert, aber es wird zuerst versucht, das Modell durch eine Anpassung der Elastizitätsmodule der Elemente zu optimieren. Welche dieser beiden Algorithmen besser geeignet sind, hängt, wie in Kapitel 4.1.3 bereits angedeutet, jeweils vom Zweck und der Fragestellung der Simulation ab.

Diese Arbeit ist zu dem Ergebnis gekommen, dass es, wie in Kapitel 2.3.1 dargelegt, aktuell genügend effektive Methoden gibt, um ein real vorliegendes Knochenstück hinreichend genau in ein FE-Programm zu transferieren. Unter Zuhilfenahme von Segmentierungsprogrammen besteht hinsichtlich der Optimierung der Netzqualität sogar die Möglichkeit, das zu simulierende Modell innerhalb eines Segmentierungsprogrammes zu vernetzen und erst dann

das fertige FE-Modell in ein FE-Programm einzulesen, um selbst kleinere Fehler bei schwierigen Modell-Formen zu vermeiden.

Eine weitere wichtige Herausforderung, für die bislang keine überzeugende Lösung gefunden wurde, ist, wie weiter oben schon angesprochen, das einfache und präzise Befunden der Randbedingungen an einem Knochen, sprich die Kräfte und Lager. Zwar gibt es mit der in-vivo-Kräftemessung beispielsweise nach Bergmann eine präzise Methode, mit der diese Aufgabe übernommen werden kann. In Anbetracht dessen, dass diese Methode später bei einer Vielzahl an Patienten möglichst nichtinvasiv und einfach ablaufen sollte, stellt sich diese Methode in der Form jedoch als nicht praktisch genug heraus. Im Rahmen dieser Arbeit konnte auch nicht genauer geklärt werden, welcher Ansatz zu einer Lösung dieses Problems beitragen kann. Die Entwicklung des Algorithmus und auch die Umsetzung der gesamten Idee hängen jedoch direkt von der Lösung dieses Problems ab.

Eine weitere auf diesem Wege ungelöste Schwierigkeit stellt die exakte Quantifizierung von nicht-mechanischen Einflüssen dar (siehe Kapitel 2.1.1 und 2.1.3), die nach dem letzten Stand der Softwaretechnik nicht in die Simulation eines FE-Programmes mit hineingerechnet werden können. Es kommt dadurch zu einer weiteren Ungenauigkeit in den Simulationen.

Nur wenn alle Voraussetzungen zur Simulation eines Modells zu einhundert Prozent quantitativ bestimmt sind, kann mit einem exakten Simulationsergebnis gerechnet werden. Bei diesen Voraussetzungen handelt es sich um den Algorithmus, ein exaktes FE-Ausgangsmodell, die genaue Einschätzung aller Randbedingungen und die genaue Einschätzung des hormonellen und sonstigen Einflusses auf den in-vivo-Knochen. Sobald auch nur eine dieser Voraussetzungen nicht genau genug erfüllt wird, ist es unerheblich, ob alle anderen Voraussetzungen genau bestimmt wurden. Das Ergebnis wird in dem Maße der Fehlerrate der einen Größe fehlerhaft sein, weil sich jeder Fehler multiplikativ auf das Gesamtergebnis auswirken wird (siehe Abbildung 21).

Eine weitere Schwierigkeit, die auch schon in Kapitel 4.1.5 angesprochen wurde, ist der hohe Rechenaufwand, der, besonders bei fein vernetzen 3-D-Modellen, hohe Rechenleistungen erfordert. Angesichts dessen, dass dieses Konzept der Knochensimulation, sowohl präzise, als auch massentauglich sein sollte und deswegen mit feinen 3-D-Modellen gerechnet werden muss, scheint es nach dem momentanen Stand der Hardwaretechnik nicht ausgeschlossen, mit sogenannten Supercomputern zu arbeiten. Diese sind in Deutschland in nahezu jeder

Universitätsstadt vorhanden und sie sind dazu in der Lage, diesen Rechenaufwand innerhalb annehmbarer Zeiten zu bewältigen.

In dieser Arbeit erwies es sich aufgrund des hohen Rechenaufwands, der bei der Simulation dreidimensionaler Knochenstücke zu erwarten ist, als notwendig heraus, vornehmlich mit zweidimensionalen Modellen zu arbeiten und auch die Zahl der Elemente auf ein Minimum zu reduzieren. Das konnte dadurch bewerkstelligt werden, dass in vorliegenden

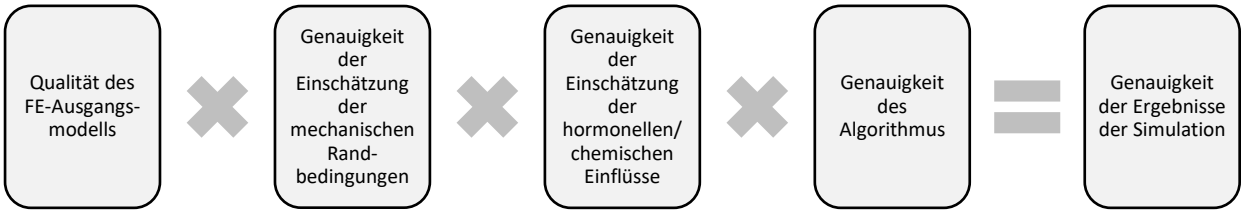


Abbildung 21: Die Genauigkeit der Ergebnisse der Simulation hängt von vielen Faktoren ab.

Knochenmodellen, zunächst von einem groben Netz ausgegangen wurde. Überall dort, wo Elemente aufgrund ihrer Spannung eindeutig entweder zum Modell gehörten oder nicht dazu gehörten, wurden diese Elemente möglichst groß gehalten und nicht weiter verfeinert. Dagegen wurde der Algorithmus so gestaltet, dass diejenigen Elemente verfeinert wurden, welche, was ihre Spannung angeht, im Grenzbereich der zu deaktivierenden Elemente lagen.

Bezüglich der Frage nach dem passenden Elementtyp muss gesagt werden, dass zwar beispielsweise die PLANE-Kategorie in ANSYS durchaus eine effektive Alternative darstellen kann. In diesem Stadium der Arbeit kann jedoch, auch wenn die Simulationen mit den entwickelten Algorithmen nicht auf etwas Gegenteiliges hinwiesen, nicht mit letzter Sicherheit gesagt werden, dass dieser Elementtyp auf jeden Fall für diese Knochen-Fragestellungen geeignet ist. Angesichts dessen, dass es für viele durchaus komplizierte Fragestellungen, wie z.B. für Holz im Programm Z88 oder für Komposite in ALGOR und NEi Nastran schon eigens Elementtypen konstruiert wurden, scheint es nicht ausgeschlossen zu sein, eine Kategorie von Elementtypen speziell für Knochen entwickeln zu können.
Die in Kapitel 4.1.4 besprochenen Lastschritte sind auch ein Baustein, der im optimalen Algorithmus zur Knochensimulation nicht fehlen darf. Unabhängig davon, welches in-vivo-Knochenstück simuliert werden soll, ist davon auszugehen, dass die realen Randbedingung in diesem Knochenstück keine statische Belastung darstellen, sondern sich situationsbedingt ändern (siehe Kapitel 2.4.1). Es ist denkbar, diesem Problem auf die Weise zu begegnen, dass

jeder Bewegungsablauf wiederum in viele einzelne Lastschritte unterteilt wird, welche dann insgesamt einer Bewegung entsprechen. Das hieße am Beispiel des Unterkiefers aus Kapitel 4.1.4, dass z.B. für die Ruheschwebelage ein Lastschritt definiert werden müsste, für die Bewegung des Sprechens weitere 100 repräsentative Lastschritte, für das Mahlen mit den Seitenzähnen 100 weitere Lastschritte und für das Abbeißen mit den Frontzähnen nochmal 100 weitere Lastschritte, insgesamt also 301 Lastschritte. Jedoch ist auch zu bedenken, dass die anteilige Anzahl der Lastschritte für jeweils eine Situation in ein und derselben Simulation ebenfalls einen großen Einfluss auf die Ergebnisse hat. Es wäre deswegen falsch, in einer Simulation das Mahlen mit den Seitenzähnen mit 100 Lastschritten zu repräsentieren und im Gegenzug die Ruheschwebelage mit nur einem Lastschritt, weil, zeitlich und deswegen auch mechanisch gesehen, in der Realität das Mahlen mit den Seitenzähnen nicht im Verhältnis 100:1 zur Ruheschwebelage steht. Diese Simulation würde aus diesem Grund zu einem falschen Ergebnis führen. Deswegen ist es wichtig, zunächst zu ermitteln, welche zeitlichen Anteile die einzelnen Lastschritte klinisch an der gesamten Simulation haben, um die Anzahl der Lastschritte dann insgesamt proportional zu definieren.

Technisch gesehen sind auch bis zu 999 Lastschritte in ANSYS definierbar. Es bleibt aber die Frage, inwiefern dieses Unterteilen der Bewegung dem Bewegungsablauf in Wirklichkeit gerecht wird. Je mehr Lastschritte pro Bewegung definiert werden, desto näher käme dies der eigentlichen Bewegung, doch es würde niemals exakt der Realität entsprechen.

Zur technischen Programmierung eines Algorithmus muss gesagt werden, dass es prinzipiell die Möglichkeit gibt, dass wenn ein Algorithmus z.B. in ANSYS in APDL-Form geschrieben wurde, und der Nutzer ihn in einem anderen FE-Programm verwenden möchte, dass dieser für das neue Programm umgeschrieben werden kann, solange in dem neuen Programm alle Bausteine, die im ursprünglichen Algorithmus verwendet wurden, auch vorzufinden sind. Solange in dem neuen Programm auch alle sonstigen Parameter, wie z.B. der Elementtyp, das Netz und die Gleichungslöser-Parameter so wie in ANSYS eingestellt wurden, sind auch in dem neuen Programm mit dem umgeschriebenen Algorithmus die gleichen oder zumindest sehr ähnliche Ergebnisse zu erwarten, wie in ANSYS.

Prinzipiell konnte gezeigt werden, dass es bei der Entwicklung des Algorithmus keine unüberwindbaren Hürden technischer Art. Das Vorliegen einer präzisen Übertragung des klinischen Modells inklusive seiner Randbedingungen im FE-Programm vorausgesetzt, sind im weiteren Ablauf alle programmiertechnischen Bausteine, die zur Entwicklung des

Algorithmus verwendet werden müssen, gegeben. Es kommt dann nur noch auf die genaue Kombination der Befehle an und auf die einzelnen zu verwendenden Werte z.B. bezüglich des an der Spannung orientierten Deaktivieren und Reaktivierens von Elementen.

5.3 Perspektive

Falls der Ansatz gelänge, über ein FE-Programm, welches mit einem Algorithmus gesteuert wird, für klinische Knochenmodelle an einem Computer Konstruktionen zu entwerfen oder Knochenentwicklungen für die Zukunft vorauszuberechnen, wäre das für klinische therapeutische Zwecke sehr zukunftsweisend. Dies brächte nicht nur bedeutsame Neuerungen für die Kieferorthopädie mit sich, sondern für die gesamte Medizin.

Ein solcher Durchbruch hieße, dass sämtliche medizinischen Implantate und auch Prothesen, deren Form einer Funktion folgt, für jeden Patienten exakt so, wie er es aufgrund seiner Physiologie benötigt, konstruiert werden könnten. Somit könnte das Risiko des vorzeitigen Versagens eines Implantates, wie es z.B. bei Hüftgelenksimplantaten häufig gesehen wird (Widjaja und Hartung 2001), reduziert werden. Bergmann wies bereits 1997 auf dieses Problem hin: „Trotz intensiver Bemühungen wurde bislang nur ein Teil der Verbesserungen der Implantate mit *ingenieurmäßigen* Methoden erreicht. Statt dessen wurde die Prothesenentwicklung weitgehend durch Erfolge und Fehlschläge beim langjährigen *klinischen* Einsatz der verschiedenartigsten Modelle bestimmt.“ (Bergmann 1997c)

Alsdann ist es auch denkbar, dass aufgrund dieser technischen Möglichkeit viele Therapieansätze, bei denen es um Knochenumstrukturierungen geht, wie z.B. das chirurgische Vor- oder Rückverlagern eines Kiefers oder die Therapie bei durch fehlenden Knochen verhindertem Durchbruch von seitlichen Inzisivi bei Lippen-Kiefer-Gaumenspalten, zum Wohle des Patienten neu überdacht und gegebenenfalls variiert werden könnten. Diesbezüglich ist es zum Beispiel denkbar, dass man im FE-Programm, aufgrund von Simulationen berechnen könnte, welche Kräfte man apparativ klinisch anbringen müsste, um diese sonst chirurgisch zu therapierenden Pathologien in Zukunft nicht invasiv anzugehen.

Letzten Endes ist die Entwicklung des perfekten Algorithmus, welcher hinreichend genaue Ergebnisse für in-vivo-Knochensimulationen liefern soll, nur Hand in Hand mit der Entwicklung einer Methode möglich ist, welche es erlaubt, hinreichend genau die

existierenden Randbedingungen eines Falles in die Simulation mit einfließen zu lassen. Denn die Verifizierung der Korrektheit dieses Algorithmus kann – wie in Kapitel 4.2 beschrieben – de facto nur durch Probesimulationen durchgeführt werden. Diese dürfen zunächst nur das Ziel haben, dass der Algorithmus lediglich den Status quo eines in-vivo-Knochens selbständig und möglichst realitätsnah simuliert. Solange es jedoch keine Methode gibt, mit der die Randbedingungen eines Falles hinreichend präzise in die FE-Simulation übertragen werden können, ist davon auszugehen, dass auch die Ergebnisse der Simulationen fehlerhaft sein werden, selbst wenn diese Fehler nicht wesentlich aus dem Algorithmus resultieren. Diese fehlerhaften Ergebnisse wiederum führen dazu, dass letztendlich keine sicheren Rückschlüsse auf die Qualität des Algorithmus möglich sind, weil nicht genau gesagt werden kann, ob etwaige Fehler aus Ungenauigkeiten bei der Angabe der Randbedingungen resultieren oder aus dem Algorithmus. Deswegen sind die Entwicklung des Algorithmus und die Entwicklung einer Methode zur exakten Quantifizierung von Randbedingungen zwei Bedingungen, welche ohne jeweils die andere nicht zu einem zufriedenstellenden Erfolg führen werden und deswegen gemeinsam vorangebracht werden müssen.

Bezüglich der Frage nach dem passenden Elementtyp ist es sinnvoll, beispielsweise in einer Zusammenarbeit zwischen Biologen, Orthopäden oder Kieferorthopäden und FE-Software-Entwicklern, zu versuchen, für ein schon bestehendes FE-Programm, wie z.B. ANSYS, die Entwicklung einer neuen Kategorie von Elementtypen voranzutreiben, die speziell nur für Knochensimulationen verwendet werden kann.

Für den Fall, dass es gelänge einen eigenen Elementtyp für in-vivo-Knochen zu konstruieren, wäre es auch denkbar, dass sich mit diesem auch die hormonellen Einflüsse und etwaige Krankheiten, wie Osteoporose, die neben den mechanischen Randbedingungen auch einen Einfluss auf einen lebenden Knochen haben, beispielsweise durch ein insgesamt herabgesetztes Elastizitätsmodul zumindest ansatzweise mit einkalkuliert werden könnten.

Letztendlich wird der Algorithmus auch – dem Wesen nach – eine Errungenschaft sein, die nicht mit letzter Sicherheit, wie eine mathematische Formel, nachgewiesen werden kann, sodass sie prinzipiell allgemeingültig wäre. Es handelt sich dabei eher um den Versuch einer Beschreibung von Veränderungsprozessen der Wirklichkeit in Programmiersprache, die, auch wenn sie schon in vielen Modellen zuverlässig funktioniert hat, sich potenziell in bestimmten Modellen später immer noch auch als falsch oder ungenau herausstellen kann und dann nachgebessert werden muss.

6. Zusammenfassung

Die Aufgabenstellung dieser Arbeit lautete, Algorithmen zu entwickeln, welche in FE-Programmen die Veränderungsprozesse lebenden Knochens berechnen können. Ausgehend von dieser Aufgabenstellung wurde deswegen zunächst diskutiert, welche Voraussetzungen für die Apposition und Resorption von in-vivo-Knochen gelten. Zu Beginn wurde aus diesem Grund auf die biologischen und physikalischen Eigenschaften von in-vivo-Knochen eingegangen. Mit Hinweis auf bisherige Versuche zur Quantifizierung von in-vivo-Muskelkräften wurde später auch festgestellt, dass es auf diesem Gebiet noch unzureichend gelöste Schwierigkeiten gibt. Anschließend wurde auf die grundsätzliche Arbeitsweise von FE-Programmen eingegangen, welche in die Abschnitte Präprozessor, Gleichungslöser, Postprozessor und dem erneutem Präprozessieren gegliedert ist. In diesem Zuge konnte auch ein kurzer Überblick über die Funktionsweise der FE-Methode gegeben werden.

Am Beispiel des FE-Programms ANSYS wurde gezeigt, dass es prinzipiell alle technischen Bausteine zur Entwicklung von Algorithmen gibt. Unter Zuhilfenahme dieser Bausteine wurden in dieser Arbeit zwei Algorithmen entwickelt. Jeder der beiden Algorithmen nutzt eine andere Kombination der vorliegenden Bausteine als der jeweils andere Algorithmus und verfolgt dadurch eine andere Strategie. Das Ziel beider Algorithmen ist aber immer die Optimierung der Form des jeweiligen FE-Modells unter den vorgegebenen Randbedingungen. Der in dieser Arbeit entwickelte Algorithmus 2 deaktiviert, reaktiviert und verfeinert gezielt Elemente um dieses Ziel zu erreichen. Insbesondere Simulationen mit der Zufallsfunktion beim Deaktivieren von Elementen mit diesem Algorithmus führten zu ersten wegweisenden Ergebnissen. Der Algorithmus 1 hingegen arbeitet primär mit der Veränderung des Elastizitätsmoduls der jeweiligen Elemente und erst sekundär mit der Deaktivierung, Reaktivierung und Verfeinerung von Elementen. Beide Algorithmen zeichnen sich unter anderem dadurch aus, dass sie mit Lastschritten arbeiten können und in mehreren Etappen und Iterationen rechnen können. Außerdem können sämtliche Parameter beispielsweise bezüglich des erneuten Präprozessierens vom Benutzer vorgegeben werden.

Die in dieser Arbeit vorliegenden Simulationen wurden mit dem Elementtyp PLANE durchgeführt. Da in dieser Arbeit nicht hinreichend genau festgestellt werden konnte, wie gut dieser Elementtyp für die spezielle Fragestellung dieser Arbeit geeignet ist, gilt die Erstellung eines eigenen Elementtyps für in-vivo-Knochen an dieser Stelle generell als empfehlenswert.

Getestet wurden die beiden Algorithmen in dieser Arbeit nur an einfachen zweidimensionalen Objekten mit einer geringen Anzahl von Kräften und Lagern. Die Resultate dieser Simulationen erscheinen aus biomechanischer Sicht zunächst plausibel. In klinischen Fallbeispielen ist jedoch davon auszugehen, dass in jeder Simulation mit einer Vielzahl von Kräften und Lagern zu rechnen ist. Um die entwickelten Algorithmen für klinische Zwecke nutzen zu können, muss also noch gezeigt werden, dass diese Algorithmen auch unter realistischeren Randbedingungen und insbesondere auch an dreidimensionalen Modellen biomechanisch plausible Ergebnisse errechnen. Diese Simulationen konnten im Rahmen dieser Arbeit jedoch nicht durchgeführt werden und werden sich daher erst in anschließenden Arbeiten zeigen können.

Schließlich wurde die Fragestellung dieser Arbeit in den Gesamtkontext angrenzender Fragestellung gesetzt. Dabei wurde festgestellt, dass die Entwicklung von speziellen Algorithmen lediglich einen notwendigen, aber keinen ausreichenden Schritt zur exakten Berechnung von klinischen Veränderungsprozessen von in-vivo-Knochen darstellt. Ohne hinreichend adäquate Methoden zur exakten und einfachen Quantifizierung der Muskelkräfte oder der hormonellen Lage eines Patienten, wird auch bei Vorliegen von akkurat arbeitenden Algorithmen keine exakte Berechnung von Knochenumbauten möglich sein. Sichere Rückschlüsse auf die Qualität der entwickelten Algorithmen lassen sich also auch erst durch ein Vorankommen in diesen angrenzenden Fragestellungen ziehen. In diesem Sinne handelt es sich bei der vorliegenden Arbeit um einen Teil eines größeren Gesamtprojektes, bei dem es mithilfe weiterer intensiver interdisziplinärer Forschung in den an dieser Arbeit angrenzenden Fragestellungen, zu einem Erfolg des Gesamtprojektes führen kann.

7. Literaturverzeichnis

Ashman RBR, J. Y. (1988). Elastic modulus of trabecular bone material. J Biomech 21. 177-179

Awang MMEM, Ibrahim Dauda (2016). Finite Element Modeling of Nanotube Structures: Linear and Non-linear Models. Switzerland: Springer

Bergmann G (1997a). In vivo Messung der Belastung von Hüftimplantaten. Berlin: Verlag Dr. Köster; S. 396.

Bergmann G (1997b). In vivo Messung der Belastung von Hüftimplantaten. Berlin: Verlag Dr. Köster; S. 102.

Bergmann G (1997c). In vivo Messung der Belastung von Hüftimplantaten. Berlin: Verlag Dr. Köster; S. 5.

Bergmann G (1997d). In vivo Messung der Belastung von Hüftimplantaten. Berlin: Verlag Dr. Köster; S. 103-9.

Bergmann G (1997e). In vivo Messung der Belastung von Hüftimplantaten. Berlin: Verlag Dr. Köster; S. 8-9.

Boryor AG, Martin; Hohmann, Ansgar; Wunderlich, Arthur;, Sander CS, Franz Martin; Sander, Franz Günter (2008). Stress distribution and displacement analysis during an intermaxillary disjunction—A three-dimensional FEM study of a human skull. J Biomech 41. 381

Brear K, Currey JD, Pond CM, Ramsay MA (1990). The mechanical properties of the femur of the Polar Bear Ursus maratimus. J Zool 222. 49-58

Carter DR (1982). Mechanical loading histories and cortical bone remodeling. Calcif Tissue Int 36. 19-24

Chan DDC, Luyao; Butz, Kent D.; Trippel, Stephen B.; Naumann, Eric A.; Neu, Corey P. (2015). In vivo articular cartilage deformation: noninvasive quantification of intratissue strain during joint contact in the human knee. Scientific Reports. 1-14

Currey JD (1988). The effect of porosity and mineral content on the young's modulus of elasticity of compact bone. J Biomech 21. 131-9

Dahl O-J, Dijkstra EW, Hoare CAR (1972). Structured Programming. London: Academic Press.

Fröhlich P (1995). FEM-Leitfaden: Einführung und praktischer Einsatz von Finite-Element-Programmen. Berlin, Heidelberg: Springer

Fröhlich P (2005). FEM-Anwendungspraxis: Einstieg in die Finite Elemente Analyse. Wiesbaden: Springer Fachmedien.

Frost HM (1987). Bone "mass" and the "mechanostat": a proposal. The Anatomical Record 219. 1-9

Fuentes ADS, Chiarella; Miralles, Rodolfo; Ferreira, Cláudia L.; Mapelli, Andrea; Lodetti, Gianluigi ; Martin, Conchita (2016). Assessment of electromyographic activity in patients with temporomandibular disorders and natural mediotrusive occlusal contact

during chewing and tooth grinding. The Journal of Craniomandibular & Sleep Practice 35. 152-160

Gross TS, Rubin CT (1995). Uniformity of resorptive bone loss induced by disuse. Journal of Orthopaedic Research 13. 708-14.

Groth P (2002). FEM-Anwendungen: Statik-, Dynamik- und Potenzialprobleme mit professioneller Software lösen. Heidelberg: Springer-Verlag.

Haapasalo H, Kontulainen S, Sievanen H, Kannus P, Jarvinen M, Vuori I (2000). Exercise-induced bone gain is due to enlargement in bone size without a change in volumetric bone density: a peripheral quantitative computed tomography study of the upper arms of male tennis players. Bone 27. 351-7

Hackenberg N (2005). Methoden zur nicht invasiven Bestimmung der in-vivo Muskelkraft in Korrelation zum Muskelquerschnitt am Beispiel der menschlichen Oberschenkelmuskulatur. München: Dissertation an der Ludwig-Maximilians-Universität. 59

Hock JM, Gera I (1992). Effects of continous and intermittent administration and inhibition of resorption of the anabolic response of bone to parathyroid hormone. Journal of Bone and Mineral Research 7. 65-72.

Hollister SJ, Brennan, J. M., and Kikuchi, N. (1994). A homogenization sampling procedure for calculating trabecular bone effective stiffness and tissue level stress. J Biomech 24. 435

Jakob F, Genest F, Baron G, Stumpf U, Rudert M, Seefried L (2015). Regulation des Knochenstoffwechsels bei Osteoporose - Neuartige Osteoporosemedikamente in der Entwicklung. Unfallchirurg 118. 925

Kojima YF, H. (2014). A finite element simulation of initial movement, orthodontic movement, and the centre of resistance of the maxillary teeth connected with an archwire. The European Journal of Orthodontics 36. 255-261.

Madenci EG, I. (2015). The Finite Element Method and Applications in Engineering Using ANSYS. New York: Springer

Mathiak FU (2010). Die Methode der finiten Elemente (FEM) - Einführung und Grundlagen. Skript. Neubrandenburg.

Meyer RB, K; Harmsen, B (1990). Computersimulation orthodontischer Zahnbewegungen. Fortschr Kieferorthop 91. 238-242

Müller GG, Clemens (2007). FEM für Praktiker - Band 1: Grundlagen. Renningen: Expert Verlag

Nasdala L (2015). FEM-Formelsammlung Statik und Dynamik. Wiesbaden: Springer

Park SC, Soo-Won; Park, Jungsoo; Han, Seung-Ho; Hong, Junghwa; Kim, Young Eun (2013). Finite element modeling to estimate the apparent material properties of trabecular bone. International Journal of Precision Engineering and Manufacturing 14. 1479

Pomberger GD, H. (2008). Algorithmen und Datenstrukturen - Eine systematische Einführung in die Programmierung. München: Pearson Studium

Regber K (2002). Auswirkungen eines Kautrainings auf die Kaumuskulatur. Freiburg: Dissertation an Albert-Ludwigs-Universität.

Reilly DTB, Albert H. (1975). The elastic and ultimate properties of compact bone tissue. J Biomech 8. 403

Rho J-YT, Ting Y.; Pharr, George M. (1997). Elastic properties of human cortical and trabecular lamellar bone measured by nanoindentation. Biomaterials 18. 1325-30

Robling AG, Burr DB, Turner CH (2000). Partitioning a daily mechanical stimulus into discrete loading bouts improves the osteogenic response to loading. Journal of Bone and Mineral Research 15. 1596-602

Rubin C, Recker R, Cullen D, Ryaby J, McCabe J, McLeod K (2004). Prevention of postmenopausal bone loss by a low-magnitude, high-frequency mechanical stimuli: a clinical trial assessing compliance, efficacy, and safety. Journal of Bone and Mineral Research 19. 343-51.

Schneider JG, M; Sander, FG (2002). Numerical experiments on long-time orthodontic tooth movement. Am J Orthod Dentofacial Orthop 121. 257-265

Schneiders DR (2016). http://www.robertschneiders.de.

Schumacher A (2013). Optimierung mechanischer Strukturen. Heidelberg: Springer Verlag.

Shahar RZ, P.; Barak, M.; riesem, A. A.; Currey, J. D.; Weiner, S. (2007). Anisotropic Poisson's ratio and compression modulus of cortical bone determined by speckle interferometry. J Biomech 40. 261

Shokrieh M (2014). Residual Stresses in Composite Materials. Woodhead Publishing Limited 48.

Skerry TM (2006). One mechanostat or many? Modifications of the site-specific response of bone to mechanical loading by nature and nurture. Journal of Musculoskeletal and Neuronal Interactions 6.122-7.

Tadeusz Stolarski YN, S. Yoshimoto (2006). Engineering Analysis with ANSYS Software. Oxford: Elsevier Butterworth-Heinemann.

Townshend PR, Rose RM, Radin, E. L. (1975). Buckling studies of single human trabeculae. Journal of Biomechanics 8.199-201.

Turner CH, Pavalko FM (1998). Mechanotransduction and functional response of the skeleton to physical stress: the mechanism and mechanics of bone adaption. Journal of Orthopaedic Science 3.346-55.

Vallaeys KKAL, H; Le Tenier, M; Hamitouche, C; Arbab-Chirani, R (2015). 3D dento-maxillary osteolytic lesion and active contour segmentation pilot study in CBCT: semi-automatic vs manual methods. Dentomaxillofacial Radiology 44. 3

Wagner W (1960). Die Versorgung der pertrochanteren Oberschenkelbrüche mit dem Rundnagel nach LEZIUS. Zbl. Chir. 85. 171

Waguespack C (2009). Mastering Autodesk Inventor 2010. Indianapolis.

Watson-Jones R (1955). Fractures and joint injuries. Edinburgh: E. & S. Livingstone Ltd.

Wichelhaus A (2000). Biomechanical investigation of torquing archwires and their risk of root resorption. Biological Mechanisms of Tooth Movement and Craniofacial Adaption, Boston, Massachusetts, USA, Harvard Society for the Advancement of Orthodontics.

Widjaja W, Hartung C (2001). Biomechanische Untersuchungen und Finite-Elemente-Analysen an einem Knochen-Implantat-Verbund. Biomed Tech 46. 351

Wikipedia-Autor (2016). https://de.wikipedia.org/wiki/Segmentierung_(Bildverarbeitung).

Wolff JH (1892). Das Gesetz der Transformation der Knochen. Hirschwald.

Zerwekh JE, Ruml LA, Gottschalk F, Pak CY (1998). The effects of twelve weeks of bed rest on bone histology, biochemical markers of bone turnover, and calcium homeostasis in eleven normal subjects. Journal of Bone and Mineral Research 13.1594-601.

8. Anhang

8.1 Algorithmus 1

Tabelle 4: Algorithmus 1

Zeile	Befehl	Erklärung
1	/SOLU	*wechselt in das Gleichungslöser-Menü, es gibt bei ANSYS drei wichtige Menüs: Präprozessor, Gleichungslöser und Postprozessor. Je nach dem welchen Befehl man ausführen will, muss man zuerst in das jeweilige Menü gehen und dann den Befehl oder die Einstellung ausführen.*
2	!*	
3	NLGEOM,1	*von hier bis Zeile 13 sind von ANSYS vorgegebene Grundeinstellungen für APDL-Skripte*
4	NROPT,FULL, ,OFF	*siehe Zeile 3*
5	STAOPT,DEFA	*siehe Zeile 3*
6	LUMPM,0	*siehe Zeile 3*
7	EQSLV, , ,0, ,DELE	*siehe Zeile 3*
8	MSAVE,0	*siehe Zeile 3*
9	PCGOPT,0, ,AUTO, , ,AUTO	*siehe Zeile 3*
10	PIVCHECK,1	*siehe Zeile 3*
11	PSTRESS,0	*siehe Zeile 3*
12	TOFFST,0,	*siehe Zeile 3*
13	!*	*siehe Zeile 3*
14		
15	*ask,E_Modul,Welches E-Modul?,17000	*stellt dem Nutzer eine Frage. Muss immer in dieser Form lauten: *ask KOMMA Variable_die_abgefragt_werden_soll KOMMA Die_eigentliche_Frage_in_Textform KOMMA Ein_Standardwert_der_genommen_werden_soll_falls_der_Nutzer_die_Frage_unbeantwortet_wegklickt*
16		
17	/PREP7	*wechselt in das Präprozessor Menü*
18	!*	*von hier bis ...*
19	!*	
20	MPDE,ALL,1	
21	TBDE,ALL,1	
22	MPTEMP,,,,,,,	*... hier gelangt man in das Menü Material Models, Material Properties, Structural, Linear, Elastic, Isotropic*
23		
24	MPTEMP,,,,,,,,	*von hier bis Zeile 27 wird der Elementtyp 1 mit einem Elastizitätsmodul von 10.000 und einer Poisson-Zahl von 0,3 definiert*
25	MPTEMP,1,0	*siehe Zeile 24*
26	MPDATA,EX,1,,10000	*siehe Zeile 24*
27	MPDATA,PRXY,1,,0.3	*siehe Zeile 24*
28		
29	MPTEMP,,,,,,,,	*von hier bis Zeile 32 wird ein Elementtyp 2 mit einem Elastizitätsmodul von 17.000 und einer Poisson-Zahl von 0,3 definiert*

Zeile	Befehl	Erklärung
30	MPTEMP,1,0	*siehe Zeile 29*
31	MPDATA,EX,2,,17000	*siehe Zeile 29*
32	MPDATA,PRXY,2,,0.3	*siehe Zeile 29*
33		
34	/SOL	*wechselt in das Gleichungslöser Menü*
35	nsel,all	*selektiert alle Knoten*
36	lsclear,all	*löscht alle Randbedingungen, die am Model anliegen (Kräfte, Lager...) Funktioniert aber nur, wenn die Knoten, an denen die Randbedingungen anliegen, vorher selektiert wurden*
37	LSSOLVE,1,n,1	*Macht je eine Rechnung für jeden Lastschritt. Wichtig ist, dass die Lastschritte vorher vom Nutzer angegeben wurden. Wenn die Lastschritte vorher vom Nutzer angegeben wurden, entspricht "n" der Anzahl der angegebenen Lastschritte. Die Zeile funktioniert folgendermaßen: LSSOLVE KOMMA Die_Nummer_des_ersten_Lastschrittes (normalerweise 1) KOMMA Die_Anzahl_der_angegebenen_Lastschritte KOMMA 1*
38	/POST1	*gelangt in das Postprozessor Menü*
39	*do,ls,1,n	*das ist eine do-Schleife, sie führt einen Befehl so häufig aus, bis die gewünschte Anzahl (in diesem Fall die Anzahl der angegebenen Lastschritte "n") erreicht ist: *do KOMMA Der_aktuelle_Wert_für_den_*do_gerade_etwas_macht (Es muss eine Variable sein. Diese Variable kann dann auch in den darauffolgenden Befehlen benutzt werden. Wenn *do 3 mal etwas machen soll, ist diese Variable im ersten Durchgang 1, dann 2 und dann 3) KOMMA von_welcher_Zahl_soll_*do_mit_der_Operation_beginnen (normalerweise 1) KOMMA wie_häufig_soll_der_Befehl_ausgeführt_werden (in diesem Fall "n"-Mal, weil wir "n" Lastschritte haben)*
40	LCDEF,ls,ls	*definiert einen Lastschrittfall: LCDEF KOMMA Der_aktuelle_Wert_für_den_*do_gerade_etwas_macht_als_Variable KOMMA Der_aktuelle_Wert_für_den_*do_gerade_etwas_macht_als_Variable*
41	*enddo	*am Ende von jedem *do kommt immer auch ein *enddo*
42	*do,ls,1,n	*siehe Zeile 39*
43	LCASE,ls	*Liest einen bestimmten Lastschritt auf das aktuelle Model ein. Das heißt er bringt alle Randbedingungen dieses Lastschrittes in das FE-Model an.*
44	*enddo	*siehe Zeile 41*
45	SUMTYPE, PRIN	*Legt die Additionsart der Ergebnisse der einzelnen Lastschritte fest, in diesem Fall Principals (Main Menu>General Postproc>Load Case>Calc Options>Stress Options)*
46	*do,ls,1,n	*siehe Zeile 39*
47	LCOPER,MIN,ls	*Führt die gewünschte Additionsart aus*
48	*enddo	*siehe Zeile 41*
49		
50	nsel,all	*selektiert alle Knoten*
51	esel,all	*selektiert alle Elemente*
52	/title,Iteration Nr. 1	*vergibt der Anzeige einen Titel: /title KOMMA titel*
53	/post1	*gelangt in das Postprozessor Menü*
54	!*	*gehört zur nächsten Zeile*

Zeile	Befehl	Erklärung
55	PLNSOL,S,3,0,1.0	*Zeigt die Ergebnisse in der knotenbasierten Lösung an: PLNSOL KOMMA S (steht für stress, also Spannungen werden angezeigt) KOMMA x_oder_y_oder_z_oder_xy_oder_yz_oder_xz_oder_1_oder_2_oder_3 (hier kann man wählen, welche Art von Spannung man angezeigt haben möchte. Man kann die shear stress in verschiedenen Achsen oder Ebenen wählen oder mit den Zahlen 1,2 oder 3 die principal stress) KOMMA 0_oder_1_oder_2 (diese drei Zahlen haben mit der Anzeige des verformten Modells zutun.) KOMMA 1.0*
56	!*	*gehört zur vorherigen Zeile*
57	*GET,stress_minimum,PLNSOL,0,Min	*findet in der knotigen Lösung des aktuellen Modells den kleinsten existierenden stress und speichert ihn in einer Variable: *get KOMMA Variable_in_der_der_Wert_gespeichert_werden_soll KOMMA PLNSOL KOMMA 0 KOMMA Min_oder_Max (Min= der Minimalwert oder der negativste Wert; Max= der Maximalwert oder der positivste Wert)*
58	*ask,Legende,Wo soll die Skala beginnen?,-0.5	
59	!*	*gehört zur nächsten Zeile*
60	/CONT,1,50,Legende,0,0	*Ändert die Farbskala in der Darstellung auf ein vom Nutzer festegelegtes Intervall und man kann die Anzahl der Farben festlegen: /CONT KOMMA 1 KOMMA 50 (Anzahl der Farben) KOMMA Legende (hier kann ein fester Wert oder so wie hier eine Variable die vorher abgefragt wurde eingetragen werden. Sie stellt die untere Grenze des Intervalls dar.) KOMMA 0 KOMMA 0 (Hier ist die obere Grenze des Intervalls. Wurde wegen dem 3rd principal stress einfach auf "0" gesetzt, da keine positiven Werte zu erwarten sind.)*
61	/REPLOT	*gehört zur vorherigen Zeile*
62	!*	*gehört zur vorherigen Zeile*
63	/UI,COPY,SAVE,png,GRAPH,COLOR,normal,portrait,yes	
64		
65	*ask,stress_grenze_a,Wo liegt die Stress-Grenze?,-0.14	*fragt nach der Variable stress_grenze_a*
66	*ask,y,Obere Grenze?,1	*fragt nach der oberen Grenze*
67	*ask,i_zahl,Wie viele Iterationen?,5	*fragt nach der Anzahl der gewünschten Iterationen*
68		
69	differenz=stress_grenze_a*0.02	*setzt die Variable differenz gleich stress_grenze_a mal 0,02*
70		
71	*do,iter,1,i_zahl,1	*alle Codes von hier bis Zeile 157 werden so häufig ausgeführt, bis die vorgegebene Anzahl an Iterationen erreicht ist*
72		
73	stress_grenze=stress_grenze_a+((iter-1)*differenz)	*setzt die Variable stress*
74		
75	esel,all	*selektiert alle Elemente*
76	*GET,Elemente_Anzahl,ELEM,0,COUNT	*ermittelt die Anzahl der Elemente*
77		
78	/post1	*gelangt in das Postprozessor Menü*

Zeile	Befehl	Erklärung
79	ETABLE,tabelle,S,3	*erstellt eine Elemente-Tabelle mit den Informationen der Elementnummern und der jeweiligen Spannung von dem Element: etable KOMMA tabelle(Name der Elemente-Tabelle) KOMMA s (steht für stress, also Spannungen werden gespeichert) KOMMA x_oder_y_oder_z_oder_xy_oder_yz_oder_xz_oder_1_oder_2_oder_3 (hier kann man wählen, welche Art von stress man gespeichert haben möchte. Man kann die shear stress in verschiedenen Achsen oder Ebenen wählen oder mit den Zahlen 1,2 oder 3 die principal stress)*
80		
81	*do,el,1,Elemente_Anzahl,1	*alle Codes von hier bis Zeile 111 werden für jedes einzelne Element so häufig ausgeführt, bis die vorgegebene Anzahl an Iterationen erreicht ist,*
82	ESEL,S,,,el	*erstellt eine Elemente-Tabelle mit den Informationen der Elementnummern und der jeweiligen Spannung von dem Element: etable KOMMA tabelle(Name der Elemente-Tabelle) KOMMA s (steht für stress, also Spannungen werden gespeichert) KOMMA x_oder_y_oder_z_oder_xy_oder_yz_oder_xz_oder_1_oder_2_oder_3 (hier kann man wählen, welche Art von stress man gespeichert haben möchte. Man kann die shear stress in verschiedenen Achsen oder Ebenen wählen oder mit den Zahlen 1,2 oder 3 die principal stress)*
83	*GET,k_or_a,elem,el,ATTR,LIVE	*stellt fest, ob das aktuelle Element aktiviert oder deaktiviert ist*
84	*if,k_or_a,eq,1,then	*alle Codes von hier bis Zeile 110 werden ausgeführt, für den Fall dass das Element aktiviert ist*
85	*GET,mat_no,mat,el	*stellt den Elementtyp fest*
86	*GET,stress,ELEM,el,ETAB,tabelle	*stellt die Belastung an diesem Element fest*
87		
88	*if,mat_no,eq,1,and,stress,le,y,then	*alle Codes von hier bis Zeile 96 werden ausgeführt, für den Fall dass das Element vom Typ 1 ist und die Belastung am Element kleiner ist als die obere Grenze*
89	/prep7	*wechselt in das Präprozessor Menü*
90	mpchg,2,el	*ändert den Elementtyp auf 2*
91	*else	*gibt an, was passieren soll, wenn die Bedingungen von Zeile 88 nicht erfüllt sind*
92	*if,mat_no,eq,2,and,stress,ge,y,then	*wenn das Element vom Typ 2 ist und die Belastung größer oder gleich der oberen Grenze ist, werden Zeile 93 und 94 ausgeführt*
93	/prep7	*wechselt in das Präprozessor Menü*
94	mpchg,1,el	*ändert den Elementtyp auf 1*
95	*endif	
96	*endif	
97		
98	*if,stress,le,stress_grenze,then	*wenn die Belastung des Elements kleiner oder gleich der Stress-Grenze ist, werden alle Befehle von hier bis Zeile 109 ausgeführt*
99	ESEL,S,,,el	*selektiert das aktuelle Element*
100	nsle,s,all	*selektiert alle Knoten, die zum selektierten Element gehören*
101	esln,a,0,all	*selektiert alle Elemente, die zu den selektierten Knoten gehören*
102	/solu	*wechselt in das Gleichungslöser Menü*
103	ealive,all	*reaktiviert die selektierten Elemente*

Zeile	Befehl	Erklärung
104	*else	*gibt an, was passieren soll, wenn die Bedingungen von Zeile 98 nicht erfüllt sind*
105	*if,stress,ge,stress_grenze,then	*wenn die Belastung des Elements größer oder gleich der Stress-Grenze ist, werden Zeile 106 und 107 ausgeführt*
106	/solu	*wechselt in das Gleichungslöser Menü*
107	ekill,el	*deaktiviert das aktuelle Element*
108	*endif	
109	*endif	
110	*endif	
111	*enddo	
112		
113	/SOL	*siehe Zeile 34 bis 48*
114	esel,s,live	*siehe Zeile 34 bis 48*
115	nsel,all	*siehe Zeile 34 bis 48*
116	lsclear,all	*siehe Zeile 34 bis 48*
117	LSSOLVE,1,n,1	*siehe Zeile 34 bis 48*
118	/POST1	*siehe Zeile 34 bis 48*
119	*do,ls,1,n	*siehe Zeile 34 bis 48*
120	LCDEF,ls,ls	*siehe Zeile 34 bis 48*
121	*enddo	*siehe Zeile 34 bis 48*
122	*do,ls,1,n	*siehe Zeile 34 bis 48*
123	LCASE,ls	*siehe Zeile 34 bis 48*
124	*enddo	*siehe Zeile 34 bis 48*
125	SUMTYPE, PRIN	*siehe Zeile 34 bis 48*
126	*do,ls,1,n	*siehe Zeile 34 bis 48*
127	LCOPER,MIN,ls	*siehe Zeile 34 bis 48*
128	*enddo	*siehe Zeile 34 bis 48*
129		
130	/title,Iteration Nr. %iter% Stress-Grenze = %stress_grenze% MpChg	*vergibt der Anzeige einen Titel: /title KOMMA titel*
131	!*	
132	PLNSOL,S,3,0,1.0	*Zeigt die Ergebnisse in der knotenbasierten Lösung an: PLNSOL KOMMA S (steht für stress, also Spannungen werden angezeigt) KOMMA x_oder_y_oder_z_oder_xy_oder_yz_oder_xz_oder_1_oder_2_oder_3 (hier kann man wählen, welche Art von Spannung man angezeigt haben möchte. Man kann die shear stress in verschiedenen Achsen oder Ebenen wählen oder mit den Zahlen 1,2 oder 3 die principal stress) KOMMA 0_oder_1_oder_2 (diese drei Zahlen haben mit der Anzeige des verformten Modells zutun.) KOMMA 1.0*
133	!*	
134		
135	!*	

Zeile	Befehl	Erklärung
136	/CONT,1,50,stress_minimum,0,0	*Ändert die Farbskala in der Darstellung auf ein vom Nutzer festegelegtes Intervall und man kann die Anzahl der Farben festlegen: /CONT KOMMA 1 KOMMA 50 (Anzahl der Farben) KOMMA Legende (hier kann ein fester Wert oder so wie hier eine Variable die vorher abgefragt wurde eingetragen werden. Sie stellt die untere Grenze des Intervalls dar.) KOMMA 0 KOMMA 0 (Hier ist die obere Grenze des Intervalls. Wurde wegen dem 3rd principal stress einfach auf "0" gesetzt, da keine positiven Werte zu erwarten sind.)*
137	/REPLOT	*gehört zur vorherigen Zeile*
138	!*	*gehört zur vorherigen Zeile*
139	/UI,COPY,SAVE,png,GRAPH,COLOR,normal,portrait,yes	*speichert die Bildschirmanzeige als Bild auf der Festplatte ab*
140		
141	EPLOT	
142	/PNUM,KP,0	
143	/PNUM,LINE,0	
144	/PNUM,AREA,0	
145	/PNUM,VOLU,0	
146	/PNUM,NODE,0	
147	/PNUM,TABN,0	
148	/PNUM,SVAL,0	
149	/NUMBER,0	
150	!*	
151	/PNUM,MAT,1	
152	/REPLOT	
153	!*	
154		
155	/UI,COPY,SAVE,png,GRAPH,COLOR,normal,portrait,yes	
156		
157	*enddo	

8.2 Algorithmus 2

Tabelle 5: Algorithmus 2

Zeile	Befehl	Erklärung
1	/SOLU	*wechselt in das Gleichungslöser-Menü, es gibt bei ANSYS drei wichtige Menüs: Präprozessor, Gleichungslöser und Postprozessor. Je nach dem welchen Befehl man ausführen will, muss man zuerst in das jeweilige Menü gehen und dann den Befehl oder die Einstellung ausführen.*
2	!*	
3	NLGEOM,1	*von hier bis Zeile 13 sind von ANSYS vorgegebene Grundeinstellungen für APDL-Skripte*
4	NROPT,FULL, ,OFF	*siehe Zeile 3*
5	STAOPT,DEFA	*siehe Zeile 3*
6	LUMPM,0	*siehe Zeile 3*
7	EQSLV, , ,0, ,DELE	*siehe Zeile 3*
8	MSAVE,0	*siehe Zeile 3*
9	PCGOPT,0, ,AUTO, , ,AUTO	*siehe Zeile 3*
10	PIVCHECK,1	*siehe Zeile 3*
11	PSTRESS,0	*siehe Zeile 3*
12	TOFFST,0,	*siehe Zeile 3*
13	!*	*siehe Zeile 3*
14		
15	*ask,E_Modul,Welcher E-Modul?,17000	*stellt dem Nutzer eine Frage. Muss immer in dieser Form lauten: *ask KOMMA Variable_die_abgefragt_werden_soll KOMMA Die_eigentliche_Frage_in_Textform KOMMA Ein_Standardwert_der_genommen_werden_soll_falls_der_Nutzer_die_Frage_unbeantwortet_wegklickt*
16		
17	/PREP7	*wechselt in das Präprozessor Menü*
18	!*	*von hier bis ...*
19	!*	
20	MPDE,ALL,1	
21	TBDE,ALL,1	
22	MPTEMP,,,,,,,	*... hier gelangt man in das Menü Material Models, Material Properties, Structural, Linear, Elastic, Isotropic*
23		
24	MPTEMP,,,,,,,,	*hier kann man ein Material Model definieren*
25	MPTEMP,1,0	
26	MPDATA,EX,1,,E_Modul	*hier wird die Nummer und der E-Modul des Material Models festgelegt: MPDATA KOMMA EX KOMMA Nummer_des_Material_Models KOMMA KOMMA Betrag_oder_Variable_des_E-Moduls (wenn man den Betrag selber festlegen möchte, kann man den einfach hier als Zahl hineinschreiben und wenn man den vorher schon als *ask abgefragt hat, muss man die Variable der Abfrage hier eintragen)*

Zeile	Befehl	Erklärung
27	MPDATA,PRXY,1,,0.3	*hier wird für das Material Model, für das in der oberen Zeile die Nummer und der E-Modul angegeben wurde, die Poissonzahl angegeben: MPDATA KOMMA PRXY KOMMA Nummer_des_Material_Models KOMMA KOMMA Der_Wert_für_die_Poissonzahl (wenn man Komma-Zahlen angeben will, dann nie mit einem Komma sondern immer mit einem Punkt)*
28		
29	/SOL	*wechselt in das Gleichungslöser Menü*
30	nsel,all	*selektiert alle Knoten*
31	lsclear,all	*löscht alle Randbedingungen, die am Model anliegen (Kräfte, Lager...) Funktioniert aber nur, wenn die Knoten, an denen die Randbedingungen anliegen, vorher selektiert wurden*
32	LSSOLVE,1,n,1	*Macht je eine Rechnung für jeden Lastschritt. Wichtig ist, dass die Lastschritte vorher vom Nutzer angegeben wurden. Wenn die Lastschritte vorher vom Nutzer angegeben wurden, entspricht "n" der Anzahl der angegebenen Lastschritte. Die Zeile funktioniert folgendermaßen: LSSOLVE KOMMA Die_Nummer_des_ersten_Lastschrittes (normalerweise 1) KOMMA Die_Anzahl_der_angegebenen_Lastschritte KOMMA 1*
33	/POST1	*gelangt in das Postprozessor Menü*
34	*do,ls,1,n	*das ist eine do-Schleife, sie führt einen Befehl so häufig aus, bis die gewünschte Anzahl (in diesem Fall die Anzahl der angegebenen Lastschritte "n") erreicht ist: *do KOMMA Der_aktuelle_Wert_für_den_*do_gerade_etwas_macht (Es muss eine Variable sein. Diese Variable kann dann auch in den darauffolgenden Befehlen benutzt werden. Wenn *do 3 mal etwas machen soll, ist diese Variable im ersten Durchgang 1, dann 2 und dann 3) KOMMA von_welcher_Zahl_soll_*do_mit_der_Operation_beginnen (normalerweise 1) KOMMA wie_häufig_soll_der_Befehl_ausgeführt_werden (in diesem Fall "n"-Mal, weil wir "n" Lastschritte haben)*
35	LCDEF,ls,ls	*definiert einen Lastschrittfall: LCDEF KOMMA Der_aktuelle_Wert_für_den_*do_gerade_etwas_macht_als_Variable KOMMA Der_aktuelle_Wert_für_den_*do_gerade_etwas_macht_als_Variable*
36	*enddo	*am Ende von jedem *do kommt immer auch ein *enddo*
37	*do,ls,1,n	*siehe Zeile 34*
38	LCASE,ls	*Liest einen bestimmten Lastschritt auf das aktuelle Model ein. Das heißt er bringt alle Randbedingungen dieses Lastschrittes in das FE-Model an.*
39	*enddo	*siehe Zeile 36*
40	SUMTYPE, PRIN	*Legt die Additionsart der Ergebnisse der einzelnen Lastschritte fest, in diesem Fall Principals (Main Menu>General Postproc>Load Case>Calc Options>Stress Options)*
41	*do,ls,1,n	*siehe Zeile 34*
42	LCOPER,MIN,ls	*Führt die gewünschte Additionsart aus*
43	*enddo	*siehe Zeile 36*
44		

Zeile	Befehl	Erklärung
45	`nsel,s,d,u`	*selektiert alle Knoten, auf denen ein Lager angegeben ist: NSEL KOMMA s_oder_r_oder_a_oder_u_oder_all_none (s=eine komplett neue Menge an Knoten selektieren, ohne die, die vorher selektiert waren. Falls kein s/r/a/u vom Nutzer angegeben wurde, wird standardmäßig immer "s" angenommen; r=aus der Menge der vorher ausgewählten Knoten eine bestimmte Menge reselektieren; a=addiert zu der vorher ausgewählten Menge eine neue Menge dazu; u=aus den vorher selektierten Knoten eine bestimmte Menge deselektieren; all=alle Knoten selektieren; none=keine Knoten selektieren) KOMMA D (steht für Lager, das kann nur in Verbindung mit einem vorherigen S/A/D/U gebraucht werden, aber nicht mit ALL/NONE) KOMMA U (steht für Lager in beliebige Richtungen x/y/z, ansonsten kann man hier auch UX/UY/UZ hineinschreiben, wenn man bestimmte Richtungen haben möchte)*
46	`nsel,a,f,f,-1000000,1000000`	*selektiert zu der vorherigen Menge eine neue Menge an Knoten dazu, auf denen bestimmte Kräfte sind: NSEL KOMMA siehe_Zeile_45 KOMMA D_für_Lager_und_F_für_Kräfte KOMMA F (bedeutet alle Kräfe in x/y/z Richtung. funktioniert nur, wenn davor auch ein f eingegeben wurde. Will man bestimmte kräfte, kann man auch fx/fy/fz eingeben.) KOMMA Untere_Grenze_des_Intervalls_in_dem_die_Kräfte_liegen_sollen (ANSYS kennt leider kein unendlich. Deswegen muss man, wenn man alle Kräfte haben will, eine große Zahl angeben) KOMMA Obere_Grenze_des_Intervalls_in_dem_die_Kräfte_liegen_sollen*
47	`ksln,s`	*selektiert die Ankerpunkte, die auf den Knoten liegen, die vorher selektiert worden waren: KSLN KOMMA S_oder_r_oder_a_oder_u (siehe_Zeile_45, analog dazu nicht Knoten sondern Ankerpunkte)*
48	`cm,kps_bc,kp`	*erstellt und speichert eine Komponente (Datenmenge) aus den selektierten Ankerpunkte und gibt dieser Komponente einen Namen: CM KOMMA Name_der_Komponente KOMMA Art_der_Menge_die_in_dieser_Komponente_gespeichert_werden_soll (volu=Volumen; area=Flächen; line=Linien; kp=Ankerpunkte; elem=Elemente; node=Knoten; wichtig ist, dass die Menge die gespeichert werden soll, zu diesem Zeitpunkt auch tatsächlich selektiert ist)*
49	`esln,s,0`	*selektiert von den im Augenblick selektierten Knoten diejenigen Elemente, die an diesen Knoten anliegen: esln KOMMA s_oder_r_oder_a_oder_u (s=eine komplett neue Menge an Elementen selektieren, ohne die, die vorher selektiert waren; r=aus der Menge der vorher ausgewählten Elemente eine bestimmte Menge reselektieren; a=addiert zu der vorher ausgewählten Menge eine neue Menge dazu; u=aus den vorher selektierten Elementen eine bestimmte Menge deselektieren) KOMMA 0_oder_1 (0=selektiert ein Element auch wenn nur eines seiner Knoten selektiert war; 1=selektiert ein Element nur, wenn alle seine Knoten selektiert waren)*

Zeile	Befehl	Erklärung
50	nsle,s	*selektiert von den im Augenblick selektierten Elementen diejenigen Knoten, die an diesen Elementen anliegen: nsle KOMMA s_oder_r_oder_a_oder_u (s=eine komplett neue Menge an Knoten selektieren, ohne die, die vorher selektiert waren; r=aus der Menge der vorher ausgewählten Knoten eine bestimmte Menge reselektieren; a=addiert zu der vorher ausgewählten Menge eine neue Menge dazu; u=aus den vorher selektierten Knoten eine bestimmte Menge deselektieren)*
51	cm,Knoten_no_kill_ref,Knoten	*siehe Zeile 48*
52	esln,a,0	*siehe Zeile 49*
53	cm,elem_no_kill_ref,elem	*siehe Zeile 48*
54		
55	nsel,all	*siehe Zeile 45*
56	esel,all	*selektiert bestimmte Elemente: ESEL KOMMA s_oder_r_oder_a_oder_u_oder_all_none (s=eine komplett neue Menge an Elementen selektieren, ohne die, die vorher selektiert waren; r=aus der Menge der vorher ausgewählten Elemente eine bestimmte Menge reselektieren; a=addiert zu der vorher ausgewählten Menge eine neue Menge dazu; u=aus den vorher selektierten Elementen eine bestimmte Menge deselektieren; all=alle Elemente selektieren; none=keine Elemente selektieren)*
57	/post1	*gelangt in das Postprozessor Menü*
58	!*	*gehört zur nächsten Zeile*
59	PLNSOL,S,3,0,1.0	*Zeigt die Ergebnisse in der knotenbasierten Lösung an: PLNSOL KOMMA S (steht für stress, also Spannungen werden angezeigt) KOMMA x_oder_y_oder_z_oder_xy_oder_yz_oder_xz_oder_1_oder_2_oder_3 (hier kann man wählen, welche Art von Spannung man angezeigt haben möchte. Man kann die shear stress in verschiedenen Achsen oder Ebenen wählen oder mit den Zahlen 1,2 oder 3 die principal stress) KOMMA 0_oder_1_oder_2 (diese drei Zahlen haben mit der Anzeige des verformten Modells zutun.) KOMMA 1.0*
60	!*	*gehört zur vorherigen Zeile*
61	*GET,stress_minimum,PLNSOL,0,Min	*findet in der knotenbasierten Lösung des aktuellen Modells den kleinsten existierenden stress und speichert ihn in einer Variable: *get KOMMA Variable_in_der_der_Wert_gespeichert_werden_soll KOMMA PLNSOL KOMMA 0 KOMMA Min_oder_Max (Min= der Minimalwert oder der negativste Wert; Max= der Maximalwert oder der positivste Wert)*
62	*ask,Legende,Wo soll die Skala beginnen?,-0.05	
63	!*	*gehört zur nächsten Zeile*

Zeile	Befehl	Erklärung
64	/CONT,1,50,Legende,0,0	*Ändert die Farbskala in der Darstellung auf ein vom Nutzer festegelegtes Intervall und man kann die Anzahl der Farben festlegen: /CONT KOMMA 1 KOMMA 50 (Anzahl der Farben) KOMMA Legende (hier kann ein fester Wert oder so wie hier eine Variable die vorher abgefragt wurde eingetragen werden. Sie stellt die untere Grenze des Intervalls dar.) KOMMA 0 KOMMA 0 (Hier ist die obere Grenze des Intervalls. Wurde wegen dem 3rd principal stress einfach auf "0" gesetzt, da keine positiven Werte zu erwarten sind.)*
65	/REPLOT	*gehört zur vorherigen Zeile*
66	!*	*gehört zur vorherigen Zeile*
67	/UI,COPY,SAVE,png,GRAPH,COLOR,reverse,portrait,yes	
68		
69	*ask,etappenzahl,Wie viele Etappen sollen berechnet werden?,3	
70		
71		
72	/prep7	*siehe Zeile 17*
73	*do,ls,1,n	*siehe Zeile 34, in dieser do-Schleife, die hier anfängt und in der Zeile 130 endet, werden von allen Lastschritte, die vom Nutzer angegeben wurden, die Randbedingungen gemäß der Ankerpunkte abgespeichert. Das muss so gemacht werden, weil wenn alle Randbedingungen auf den Knoten angegeben wurden und die Knotennummern sich beim Verfeinern ändern, müssen die Kräfte wieder auf dieselben Stellen angebracht werden. Das geht nur, wenn man die Stellen über feste Ankerpunkte wiederfindet. Deswegen werden hier alle Kräfte und Lager anhand der Ankerpunkte in der Datei abgespeichert. in einer späteren Schleife werden diese Kräfte und Lager wieder auf die neuen Knoten an den Stellen der Ankerpunkte wieder angebracht.*
74		
75	nsel,all	*siehe Zeile 45*
76	lsclear,all	*siehe Zeile 31*
77	lsread,ls	*liest einen bestimmten Lastschritt auf das Modell ein, das heißt er bringt die Randbedingungen an: lsread KOMMA ls (Nummer des einzulesenden Lastschritts oder entsprechend eine Variable)*
78		
79	nsel,s,d,ux	*siehe Zeile 45*
80	ksln	*siehe Zeile 47*
81	cm,kpdx%ls%,kp	*siehe Zeile 48, nur mit dem Unterschied dass in dem Komponentennamen eine Variable eingefügt wurde. Hat man, so wie hier, keinen festen Komponentennamen, sondern einen Namen, der sich mit einer Variablen ändern soll (z.B. für NameX: Name1, Name2, Name3...), kann man zunächst den Namen normal eingeben und dann den Variablen mit einem % links und einem % rechts angeben. Hier ist das also kpdx%ls% -> kpdx1, kpdx2 ... je nach dem, was ls gerade ist.*

Zeile	Befehl	Erklärung
82		
83	nsel,s,d,uy	*siehe Zeile 45*
84	ksln	*siehe Zeile 47*
85	cm,kpdy%ls%,kp	*siehe Zeile 81*
86		
87	nsel,s,d,uz	*siehe Zeile 45*
88	ksln	*siehe Zeile 47*
89	cm,kpdz%ls%,kp	*siehe Zeile 81*
90		
91	nsel,s,f,fx,-1000000,1000000	*siehe Zeile 46*
92	ksln	*siehe Zeile 47*
93	*GET,kpx_Anzahl%ls%,kp,0,Count	*findet die Anzahl der selektierten Ankerpunkte heraus und speichert sie in einer Variable: *GET KOMMA kpx_Anzahl%ls% (Name der Variable, bitte Erklärung zum Variablennamen aus Zeile 81 beachten) KOMMA kp_oder_node_oder_elem_oder_line_oder_area_oder_volu (entspricht der Einheit, dessen Anzahl man herausfinden will. Hier geht es um Ankerpunkte, also KP) KOMMA 0 (muss immer so sein) KOMMA Count*
94	*DIM,fx_Array%ls%,Array,kpx_Anzahl%ls%,2	
95	*VGET,fx_Array%ls%,kp,,klist	*füllt die erste Spalte des Arrays mit den Nummern der selektierten Ankerpunkte aus: *VGET KOMMA Name_des_Array (Namenserklärungen aus Zeile 81 beachten) KOMMA kp_oder_node_oder_elem_oder_line_oder_area_oder_volu (entspricht der Einheit, dessen Nummern man in das Array füllen will. Hier geht es um Ankerpunkte, also KP) KOMMA KOMMA klist_oder_nlist_oder_elist_oder_vlist_oder_alist_oder_llist (je nach dem welche Einheit man ausgesucht hat, bekommt man entsprechend eine Liste der selektierten Einheiten des Typs)*
96	*do,aa,1,kpx_Anzahl%ls%	
97	ksel,s,kp,,fx_Array%ls%(aa,1)	*selektiert einen Ankerpunkt gemäß der Nummern in der ersten Spalte des Arrays: ksel KOMMA s (neues Set von Ankerpunkte selektieren) KOMMA kp KOMMA KOMMA fx_Array%ls%(aa,1) (findet die Stelle im Array. aa ist dabei die Position der Zeile. Wenn aa 3 ist, bedeutet das also Spalte 1, Zeile 3. Dann wird geschaut, welche Ankerpunktnummer in Spalte 1, Zeile 3 sich befindet und es wird gemäß dieser Ankerpunktnummer dann der entsprechende Ankerpunkt selektiert)*
98	nslk	*selektiert von den selektierten Ankerpunkten die Knoten, die auf derselben Stelle liegen*
99	n_1=NDNEXT(-1)	*die Knotennummer von dem Knoten, der in der oberen Zeile ausgesucht wurde, bekommt in dieser Zeile den Namen n_1*
100	*GET,ax,Knoten,n_1,f,fx	*hier wird die Kraft in x-Richtung, die an dem Knoten n_1 anliegt, unter einem bestimmten Namen gespeichert: *GET KOMMA ax (Name, unter dem der Betrag der Kraft gespeichert werden soll) KOMMA Knoten(es soll nach Knoten geschaut werden) KOMMA n_1 (Nummer oder Name des Knoten, nach dem geschaut werden soll) KOMMA f(es soll nach der Kraft geschaut werden) KOMMA fx (Kraft in xRichtung, es gibt auch fy und fz)*

Zeile	Befehl	Erklärung
101	*vfill,fx_Array%ls%(aa,2),data,ax	*hier wird die zweite Spalte des Arrays jeweils mit dem Betrag der kraft, die an dem Knoten (dessen Ankerpunktnummer in der ersten Spalte steht) anliegt, eingetragen: *vfill KOMMA fx_Array%ls%(aa,2)(Welche Stelle im Array soll gefüllt werden? Array_Name Klammer_auf Zeilenzahl_oder_entsprechende_Variable KOMMA Spaltenzahl Klammer_zu) KOMMA data(muss immer so sein) KOMMA ax(mit welchem Wert soll die angegebene Stelle im Array gefüllt werden? Konkreten Wert oder entsprechende Variable angeben)*
102	*enddo	*Ende der do-Schleife*
103		
104	nsel,s,f,fy,-1000000,1000000	*siehe Zeile 46*
105	ksln	*siehe Zeile 47*
106	*GET,kpy_Anzahl%ls%,kp,0,Count	*siehe Zeile 93*
107	*DIM,fy_Array%ls%,Array,kpy_Anzahl%ls%,2	
108	*VGET,fy_Array%ls%,kp,,klist	*siehe Zeile 95*
109	*do,aa,1,kpy_Anzahl%ls%	*siehe Zeile 96*
110	ksel,s,kp,,fy_Array%ls%(aa,1)	*siehe Zeile 97*
111	nslk	*siehe Zeile 98*
112	n_1=NDNEXT(-1)	*siehe Zeile 99*
113	*GET,ay,Knoten,n_1,f,fy	*siehe Zeile 100*
114	*vfill,fy_Array%ls%(aa,2),data,ay	*siehe Zeile 101*
115	*enddo	*siehe Zeile 102*
116		
117	nsel,s,f,fz,-1000000,1000000	*siehe Zeile 46*
118	ksln	*siehe Zeile 47*
119	*GET,kpz_Anzahl%ls%,kp,0,Count	*siehe Zeile 93*
120	*DIM,fz_Array%ls%,Array,kpz_Anzahl%ls%,2	
121	*VGET,fz_Array%ls%,kp,,klist	*siehe Zeile 95*
122	*do,aa,1,kpz_Anzahl%ls%	*siehe Zeile 96*
123	ksel,s,kp,,fz_Array%ls%(aa,1)	*siehe Zeile 97*
124	nslk	*siehe Zeile 98*
125	n_1=NDNEXT(-1)	*siehe Zeile 99*
126	*GET,az,Knoten,n_1,f,fz	*siehe Zeile 100*
127	*vfill,fz_Array%ls%(aa,2),data,az	*siehe Zeile 101*
128	*enddo	*siehe Zeile 102*
129		
130	*enddo	*Ende der großen do-Schleife. Jetzt wurden alle Randbedingungen von allen Loadsteps gemäß den entsprechenden Ankerpunkte in der Datei abgespeichert.*

Zeile	Befehl	Erklärung
131		
132		
133	*do,etappe,1,etappenzahl,1	*siehe Zeile 34, Hier beginnt der eigentliche Algorithmus. Er besteht aus mehreren do-Schleifen, die ineinander liegen. Die do-Schleife, die hier beginnt, lässt den Algorithmus, der in ihm liegt mehrmals ausführen, also so oft der Nutzer es in Zeile 69 angegeben hat.*
134		
135	/prep7	*siehe Zeile 17*
136	*do,ls,1,n	*siehe Zeile 34: In dieser do-Schleife (endet in Zeile 175) werden alle Randbedingungen aller Lastschritte auf ein Mal auf das Modell aufgetragen. Das dient dem Zweck, dass der Nutzer einen ersten Eindruck vom Ergebnis der Simulation bekommt und anschließend angeben kann, wie er die Legende haben möchte, in welchem Intervall und wie viel Prozent er killen möchte usw.*
137		
138	nsel,all	*siehe Zeile 30*
139	ksel,all	*siehe Zeile 30, analog dazu Ankerpunkte und nicht Knoten*
140	lsclear,all	*siehe Zeile 31*
141		
142	*do,aa,1,kpx_Anzahl%ls%	*siehe Zeile 34, diese do-Schleife legt alle Kräfte in x-Richtung auf die entsprechenden Knoten an. Später geschieht das gleiche für die y und z Richtung*
143	ksel,s,kp,,fx_Array%ls%(aa,1)	*siehe Zeile 97*
144	nslk	*siehe Zeile 98*
145	f,all,fx,fx_Array%ls%(aa,2)	*trägt eine bestimmte Kraft auf bestimmte Einheiten auf: f KOMMA all (die Knoten, auf denen die kraft angebracht werden soll, in diesem fall alle) KOMMA fx_oder_fy_oder_fz (in welche Richtung soll die Kraft sein?) KOMMA fx_Array%ls%(aa,2) (wie groß ist die Kraft? In diesem Fall soll der Betrag der Kraft aus dem Array aus einer bestimmten Zeile und Spalte herausgesucht werden, siehe Bemerkungen in Zeile 101)*
146	*enddo	*Ende der Schleife, die in Zeile 142 begonnen hat*
147		
148	*do,aa,1,kpy_Anzahl%ls%	*siehe Zeile 142*
149	ksel,s,kp,,fy_Array%ls%(aa,1)	*siehe Zeile 97*
150	nslk	*siehe Zeile 98*
151	f,all,fy,fy_Array%ls%(aa,2)	*siehe Zeile 145*
152	*enddo	*Ende der Schleife, die in Zeile 148 begonnen hat*
153		
154	*do,aa,1,kpz_Anzahl%ls%	*siehe Zeile 142*
155	ksel,s,kp,,fz_Array%ls%(aa,1)	*siehe Zeile 97*
156	nslk	*siehe Zeile 98*
157	f,all,fz,fz_Array%ls%(aa,2)	*siehe Zeile 145*
158	*enddo	*Ende der Schleife, die in Zeile 154 begonnen hat*

Zeile	Befehl	Erklärung
159		
160	`cmsel,s,kpdx%ls%,kp`	*selektiert eine bestimmte Menge, die vorher als Komponente gespeichert worden ist: cmsel KOMMA s_oder_r_oder_a_oder_u (s=eine komplett neue Menge selektieren, ohne das, was vorher selektiert war; r=aus der Menge der vorher selektierten Menge eine bestimmte Teilmenge selektieren; a=addiert zu der vorher ausgewählten Menge eine neue Menge dazu; u=aus den vorher selektierten Menge eine bestimmte Teilmenge deselektieren) KOMMA kpdx%ls%(Name der Komponente dessen Einheiten selektiert werden sollen, Namenserklärungen aus Zeile 81 beachten) KOMMA kp(Art der Menge, die selektiert werden soll. kp=Ankerpunkte; node=Knoten; elem=Elemente; line=Linien; area=Fläche; volu=Volumen)*
161	`nslk`	*siehe Zeile 98*
162	`D,all,,,,,,UX,,,,,`	*bringt eine Lagerstelle an selektierten Knoten an: D,all,,,,,,UX,,,,,(für alle selektierten Knoten in x-Richtung eine Lagerung anbringen. um nicht an allen selektierten Knoten, sondern nur an einem bestimmten diese Aktion vorzunehmen muss man statt all die Knotennummer von dem jeweiligen Knoten eintippen)*
163		
164	`cmsel,s,kpdy%ls%,kp`	*selektiert eine bestimmte Menge, die vorher als Komponente gespeichert worden ist: cmsel KOMMA s_oder_r_oder_a_oder_u (s=eine komplett neue Menge selektieren, ohne das, was vorher selektiert war; r=aus der Menge der vorher selektierten Menge eine bestimmte Teilmenge selektieren; a=addiert zu der vorher ausgewählten Menge eine neue Menge dazu; u=aus den vorher selektierten Menge eine bestimmte Teilmenge deselektieren) KOMMA kpdx%ls%(Name der Komponente dessen Einheiten selektiert werden sollen, Namenserklärungen aus Zeile 81 beachten) KOMMA kp (Art der Menge, die selektiert werden soll. kp=Ankerpunkte; node=Knoten; elem=Elemente; line=Linien; area=Fläche; volu=Volumen)*
165	`nslk`	*siehe Zeile 98*
166	`D,all,,,,,,,UY,,,,`	*bringt eine Lagerstelle an selektierten Knoten an: D,all,,,,,,,UY,,,, (für alle selektierten Knoten in y-Richtung eine Lagerung anbringen. Um nicht an allen selektierten Knoten, sondern nur an einem bestimmten, diese Aktion vorzunehmen, muss man, statt all die Knotennummer von dem jeweiligen Knoten eintippen)*
167		
168	`cmsel,s,kpdz%ls%,kp`	*selektiert eine bestimmte Menge, die vorher als Komponente gespeichert worden ist: cmsel KOMMA s_oder_r_oder_a_oder_u (s=eine komplett neue Menge selektieren, ohne das, was vorher selektiert war; r=aus der Menge der vorher selektierten Menge eine bestimmte Teilmenge selektieren; a=addiert zu der vorher ausgewählten Menge eine neue Menge dazu; u=aus den vorher selektierten Menge eine bestimmte Teilmenge deselektieren) KOMMA kpdx%ls% (Name der Komponente dessen Einheiten selektiert werden sollen, Namenserklärungen aus Zeile 81 beachten) KOMMA kp (Art der Menge, die selektiert werden soll. kp=Ankerpunkte; node=Knoten; elem=Elemente; line=Linien; area=Fläche; volu=Volumen)*

Zeile	Befehl	Erklärung
169	nslk	*siehe Zeile 98*
170	D,all,,,,,,,,UZ,,,	*bringt eine Lagerstelle an selektierten Knoten an: D,all,,,,,,,,UZ,,,(für alle selektierten Knoten in z-Richtung eine Lagerung anbringen. um nicht an allen selektierten Knoten, sondern nur an einem bestimmten diese Aktion vorzunehmen muss man statt all die Knotennummer von dem jeweiligen Knoten eintippen)*
171		
172	nsel,all	*siehe Zeile 45*
173	lswrite,ls	*speichert die aktuell angebrachten Randbedingungen als einen Lastschritt mit einer bestimmten Nummer ab: lswrite KOMMA ls (Nummer oder Variable unter der der Lastschritt abgespeichert werden soll)*
174		
175	*enddo	*bezieht sich auf Zeile 136*
176		
177	esel,s,live	*selektiert alle lebenden (nicht deaktivierten Elemente: ESEL KOMMA s_oder_r_oder_a_oder_u_oder_all_none (s=eine komplett neue Menge an Elementen selektieren, ohne die, die vorher selektiert waren; r=aus der Menge der vorher ausgewählten Elemente eine bestimmte Menge reselektieren; a=addiert zu der vorher ausgewählten Menge eine neue Menge dazu; u=aus den vorher selektierten Elementen eine bestimmte Menge deselektieren; all=alle Elemente selektieren; none=keine Elemente selektieren) KOMMA live (kann nur verwendet werden, wenn vorher ein "s" angegeben wurde, bezeichnet die nicht-gekillten Elemente)*
178	/SOL	*siehe Zeile 29*
179	nsel,all	*siehe Zeile 30*
180	lsclear,all	*siehe Zeile 31*
181	LSSOLVE,1,n,1	*siehe Zeile 32*
182	/POST1	*siehe Zeile 33*
183	*do,ls,1,n	*siehe Zeile 34*
184	LCDEF,ls,ls	*siehe Zeile 35*
185	*enddo	*siehe Zeile 36*
186	*do,ls,1,n	*siehe Zeile 37*
187	LCASE,ls	*siehe Zeile 38*
188	*enddo	*siehe Zeile 39*
189	SUMTYPE, PRIN	*siehe Zeile 40*
190	*do,ls,1,n	*siehe Zeile 41*
191	LCOPER,MIN,ls	*siehe Zeile 42*
192	*enddo	*siehe Zeile 43*
193		
194	esel,s,live	*siehe Zeile 177*
195	eplot	*alle selektierten Elemente anzeigen*
196	/post1	*siehe Zeile 33*

Zeile	Befehl	Erklärung
197	/title,Etappe %etappe%	*Beschriftung für die Anzeige: /title KOMMA Etappe %etappe%(Test der Beschriftung, Namenserklärungen aus Zeile 81 beachten)*
198	!*	*siehe Zeile 58*
199	PLNSOL,S,3,0,1.0	*siehe Zeile 58*
200	!*	*siehe Zeile 58*
201		
202	*if,etappe,gt,1,then	*Eine Wenn-funktion: Wenn eine bestimmte Bedingung erfüllt ist, soll ein bestimmter Befehl ausgeführt werden. Braucht immer auch am ende ein *endif. *if KOMMA etappe (das ist die Zahl oder Variable die verglichen werden soll),gt_oder_lt_oder_eq_oder_le_oder_ge (Was soll geprüft werden? gt=größer als; lt=kleiner als; eq=gleich; le=kleiner oder gleich; ge=größer oder gleich) KOMMA 1 (die Variable oder Zahl mit der die erste verglichen werden soll) KOMMA then (nach dem then kommt der Befehl, der ausgeführt werden soll, falls der Vergleich zutrifft)*
203	*ask,Legende_%etappe%,Wo soll die Skala beginnen?,-0.05	
204	!*	*siehe Zeile 63*
205	/CONT,1,50,Legende_%etappe%,0,0	*siehe Zeile 64*
206	/REPLOT	*siehe Zeile 65*
207	!*	*siehe Zeile 66*
208	*endif	*bezieht sich auf das *if von Zeile 202*
209		
210	*GET,stress_minimum_%etappe%,PLNSOL,0,Min	
211	*ask,stress_grenze_%etappe%,Wo liegt die Stress-Grenze?,-0.004	
212	*ask,s_1_%etappe%,Wo liegt die 2. Kill-Grenze?,-0.007	
213	*ask,s_1prozent_%etappe%,Wo viel Prozent sollen gekillt werden (ganze Zahl)?,10	
214	*ask,s_2_%etappe%,Wo liegt die 3. Kill-Grenze?,-0.01	
215	*ask,s_2prozent_%etappe%,Wo viel Prozent sollen gekillt werden (ganze Zahl)?,5	
216	*ask,i_zahl_%etappe%,Wie viele Iterationen in dieser Etappe?,2	
217	*ask,ref_freq_%etappe%,Nach wie vielen Iterationen soll immer verfeinert werden?,10	
218	*ask,ref_wieoft_%etappe%,Wie oft soll insgesamt verfeinert werden?,1	

Zeile	Befehl	Erklärung
219	ref_var_%etappe%=1	*hier wird eine Variable festgelegt. Namenserklärungen aus Zeile 81 beachten, ref_var_%etappe% soll ab jetzt gleich 1 sein. Allgemein kann so immer eine Variable festgelegt werden. Variable Gleichheitszeichen Zahl_oder_andere_Variable*
220		
221		
222	*do,iter,1,i_zahl_%etappe%,1	*siehe Zeile 34, Namenserklärungen aus Zeile 81 beachten, in den vorherigen Zeilen wurden sämtliche werte abgefragt, nach denen der Algorithmus agieren soll. Ab hier beginnt der Hauptteil des Algorithmus. Er erstreckt sich bis Zeile 443*
223		
224	/prep7	*siehe Zeile 17*
225	*do,ls,1,n	*siehe Zeile 34, von hier bis Zeile 264 werden für jede Iteration sämtliche Lastschritte, die am Anfang vom Nutzer angegeben worden waren, neu abgespeichert, weil es ja sein könnte, dass sich die Knotennummern im Verlauf der Iteration geändert haben und deshalb die Lastschritte auf Basis der alten Knotennummern nicht mehr gültig sind. Im Prinzip läuft das hier genau so ab wie in der do-Schleife von Zeile 136 bis Zeile 175.*
226		
227	nsel,all	*siehe Zeile 30*
228	ksel,all	*siehe Zeile 30, analog dazu Ankerpunkte und nicht Knoten*
229	lsclear,all	*siehe Zeile 31*
230		
231	*do,aa,1,kpx_Anzahl%ls%	*siehe Zeile 34, diese do-Schleife legt alle Kräfte in x-Richtung auf die entsprechenden Knoten an. Später geschieht das gleiche für die y und z Richtung*
232	ksel,s,kp,,fx_Array%ls%(aa,1)	*siehe Zeile 97*
233	nslk	*siehe Zeile 98*
234	f,all,fx,fx_Array%ls%(aa,2)	*trägt eine bestimmte Kraft auf bestimmte Einheiten auf: f KOMMA all (die Knoten, auf denen die kraft angebracht werden soll, in diesem Fall alle) KOMMA fx_oder_fy_oder_fz (in welche Richtung soll die kraft sein?) KOMMA fx_Array%ls%(aa,2) (wie groß ist die Kraft? In diesem Fall soll der Betrag der Kraft aus dem Array aus einer bestimmten Zeile und Spalte herausgesucht werden, siehe Bemerkungen in Zeile 101)*
235	*enddo	*Ende der Schleife, die in Zeile 231 begonnen hat*
236		
237	*do,aa,1,kpy_Anzahl%ls%	*siehe Zeile 142*
238	ksel,s,kp,,fy_Array%ls%(aa,1)	*siehe Zeile 97*
239	nslk	*siehe Zeile 98*
240	f,all,fy,fy_Array%ls%(aa,2)	*siehe Zeile 145*
241	*enddo	*Ende der Schleife, die in Zeile 237 begonnen hat*
242		
243	*do,aa,1,kpz_Anzahl%ls%	*siehe Zeile 142*

Zeile	Befehl	Erklärung
244	ksel,s,kp,,fz_Array%ls%(aa,1)	*siehe Zeile 97*
245	nslk	*siehe Zeile 98*
246	f,all,fz,fz_Array%ls%(aa,2)	*siehe Zeile 145*
247	*enddo	*Ende der Schleife, die in Zeile 243 begonnen hat*
248		
249	cmsel,s,kpdx%ls%,kp	*siehe Zeile 160*
250	nslk	*siehe Zeile 98*
251	D,all,,,,,,UX,,,,,	*siehe Zeile 162*
252		
253	cmsel,s,kpdy%ls%,kp	*siehe Zeile 164*
254	nslk	*siehe Zeile 98*
255	D,all,,,,,,,UY,,,,	*siehe Zeile 166*
256		
257	cmsel,s,kpdz%ls%,kp	*siehe Zeile 168*
258	nslk	*siehe Zeile 98*
259	D,all,,,,,,,,UZ,,,	*siehe Zeile 170*
260		
261	nsel,all	*siehe Zeile 30*
262	lswrite,ls	*siehe Zeile 173*
263		
264	*enddo	*Ende der Schleife die in Zeile 225 begonnen hat*
265		
266	esel,s,live	*siehe Zeile 177*
267	/SOL	*siehe Zeile 1*
268	nsel,all	*siehe Zeile 30*
269	lsclear,all	*siehe Zeile 31*
270	LSSOLVE,1,n,1	*siehe Zeile 32*
271	/POST1	*siehe Zeile 33*
272	*do,ls,1,n	*siehe Zeile 34*
273	LCDEF,ls,ls	*siehe Zeile 35*
274	*enddo	*siehe Zeile 36*
275	*do,ls,1,n	*siehe Zeile 37*
276	LCASE,ls	*siehe Zeile 38*
277	*enddo	*siehe Zeile 39*
278	SUMTYPE, PRIN	*siehe Zeile 40*
279	*do,ls,1,n	*siehe Zeile 41*
280	LCOPER,MIN,ls	*siehe Zeile 42*
281	*enddo	*siehe Zeile 43*
282		
283	/post1	*siehe Zeile 57*
284	nsel,all	*siehe Zeile 30*

Zeile	Befehl	Erklärung
285	etable,tabelle,s,3	*erstellt eine Elemente-Tabelle mit den Informationen der Elementnummern und der jeweiligen Spannung von dem Element: etable KOMMA tabelle(Name der Elemente-Tabelle) KOMMA s (steht für stress, also Spannungen werden gespeichert) KOMMA x_oder_y_oder_z_oder_xy_oder_yz_oder_xz_oder_1_oder_2_oder_3 (hier kann man wählen, welche Art von stress man gespeichert haben möchte. Man kann die shear stress in verschiedenen Achsen oder Ebenen wählen oder mit den Zahlen 1,2 oder 3 die principal stress)*
286	ESEL,S,ETAB,TABELLE,s_1_%etappe%,stress_grenze_%etappe%, ,0	
287	*GET,e_zahl,elem,0,count	*siehe Zeile 93, entsprechend geht es hier um Elemente*
288	*if,e_zahl,gt,0,then	*siehe Zeile 202, in diesem *if soll geprüft werden, ob e_zahl größer 0 ist. Das heißt es wird geschaut, ob zwei Zeilen vorher überhaupt irgendwelche Elemente selektiert wurden. Wenn ja, dann werden die folgenden Befehle ausgeführt, wenn nicht, dann nicht.*
289	*GET,DIM,ACTIVE,0,TIME,WALL	*Zufallsgenerator*
290	dim=dim*3600	*Zufallsgenerator*
291	*DIM,dtab,Array,e_zahl,2	*siehe Zeile 94*
292	*dim,altArray,,e_zahl	*siehe Zeile 94*
293	*VGET,dtab(1,1),elem,,elist	*siehe Zeile 95, nach dem *vget wird hier nach einem bestimmten Wert in einem Array geschaut. Das geht folgendermaßen: Arrayname Klammer_auf Zeilenzahl_der_Variable_die_man_haben_will KOMMA Spaltenzahl_der_Variable_die_man_haben_will Klammer_zu, das kann man generell immer so machen, wenn irgendwo nach einer zahl gefragt ist, dass man dann statt der Zahl einfach eine bestimmte Stelle in einem Array nennt, damit er diese Zahl nimmt.*
294	*vfill,dtab(1,2),rand,0,1	*siehe Zeile 101, siehe Array-Bemerkungen aus Zeile 293*
295	*moper,altArray(1),dtab(1,1),sort,dtab(1,2)	
296	esel,none	*siehe Zeile 56*
297	*do,i,1,nint(e_zahl*s_2prozent_%etappe%*0.01)	
298	esel,a,elem,,dtab(i,1)	*siehe Zeile 56*
299	*enddo	*bezieht sich auf das *do von Zeile 297*
300	*del,altArray	*löscht ein Array, das vorher erstellt worden war: *del KOMMA altArray (Name des Arrays, das gelöscht werden soll)*
301	*del,dtab	*siehe Zeile 300*
302	CM,Komponente1,elem	*siehe Zeile 48, entsprechend für Elemente*
303	*endif	*bezieht sich auf das *if aus Zeile 288*
304		
305	nsel,all	*siehe Zeile 30*
306	ESEL,S,ETAB,TABELLE,s_2_%etappe%,s_1_%etappe%,,0	

Zeile	Befehl	Erklärung
307	*GET,e_zahl,elem,0,count	*siehe Zeile 93, entsprechend geht es hier um Elemente*
308	*if,e_zahl,gt,0,then	*siehe Zeile 288*
309	*GET,DIM,ACTIVE,0,TIME,WALL	*siehe Zeile 289*
310	dim=dim*3600	*siehe Zeile 290*
311	*DIM,dtab,Array,e_zahl,2	*siehe Zeile 94*
312	*dim,altArray,,e_zahl	*siehe Zeile 94*
313	*VGET,dtab(1,1),elem,,elist	*siehe Zeile 95, nach dem *vget wird hier nach einem bestimmten Wert in einem Array geschaut. Das geht folgendermaßen: Arrayname Klammer_auf Zeilenzahl_der_Variable_die_man_haben_will KOMMA Spaltenzahl_der_Variable_die_man_haben_will Klammer_zu, das kann man generell immer so machen, wenn irgendwo nach einer Zahl gefragt ist, dass man dann statt der Zahl einfach eine bestimmte Stelle in einem Array nennt, damit er diese Zahl nimmt.*
314	*vfill,dtab(1,2),rand,0,1	*siehe Zeile 101, siehe Array-Bemerkungen aus Zeile, 293*
315	*moper,altArray(1),dtab(1,1),sort,dtab(1,2)	
316	esel,none	*siehe Zeile 56*
317	*do,i,1,nint(e_zahl*s_1prozent_%etappe%*0.01)	
318	esel,a,elem,,dtab(i,1)	*siehe Zeile 56*
319	*enddo	*bezieht sich auf das *do von Zeile 317*
320	*del,altArray	*siehe Zeile 300*
321	*del,dtab	*siehe Zeile 300*
322	CM,Komponente2,elem	*siehe Zeile 48, entsprechend für Elemente*
323	*endif	*bezieht sich auf das *if aus Zeile 308*
324		
325	nsel,all	*siehe Zeile 30*
326	ESEL,S,ETAB,TABELLE,stress_grenze_%etappe%,1000000, ,0	
327	cm,komponente0,elem	*siehe Zeile 48, entsprechend für Elemente*
328		
329	nsel,all	*siehe Zeile 30*
330	ESEL,S,ETAB,TABELLE,stress_minimum_%etappe%*1000,s_2_%etappe%, ,0	
331	cm,komponente3,elem	*siehe Zeile 48, entsprechend für Elemente*
332	cmsel,s,komponente1	*siehe Zeile 160*
333	cmsel,a,komponente2	*siehe Zeile 160*
334	cmsel,a,komponente0	*siehe Zeile 160*
335	cmsel,u,komponente3	*siehe Zeile 160*
336	cmsel,u,elem_no_kill_ref	*siehe Zeile 160*
337		

Zeile	Befehl	Erklärung
338	/solu	*siehe Zeile 1*
339	ekill,all	*Löscht bestimmte Elemente aus: ekill KOMMA ALL (All=alle selektierten Elemente werden gekillt, falls man nichts eingibt, wird all als Standard angenommen; will man nur ein bestimmtes Element killen, kann man hier ansonsten die entsprechende Elementnummer angeben)*
340	esel,s,live	*siehe Zeile 177*
341	nsel,all	*siehe Zeile 30*
342	lsclear,all	*siehe Zeile 31*
343	LSSOLVE,1,n,1	*siehe Zeile 32*
344	/POST1	*siehe Zeile 33*
345	*do,ls,1,n	*siehe Zeile 34*
346	LCDEF,ls,ls	*siehe Zeile 35*
347	*enddo	*siehe Zeile 36*
348	*do,ls,1,n	*siehe Zeile 37*
349	LCASE,ls	*siehe Zeile 38*
350	*enddo	*siehe Zeile 39*
351	SUMTYPE, PRIN	*siehe Zeile 40*
352	*do,ls,1,n	*siehe Zeile 41*
353	LCOPER,MIN,ls	*siehe Zeile 42*
354	*enddo	*siehe Zeile 43*
355	/title,Etappe %etappe%, Iteration Nr. %iter%, Kill	
356	!*	*siehe Zeile 58*
357	PLNSOL,S,3,0,1.0	*siehe Zeile 59*
358	!*	*siehe Zeile 60*
359		
360	!*	*siehe Zeile 63*
361	/CONT,1,50,stress_minimum_%etappe%,0,0	*siehe Zeile 64*
362	/REPLOT	*siehe Zeile 65*
363	!*	*siehe Zeile 66*
364		
365	nsel,all	*siehe Zeile 30*
366	etable,tabelle,s,3	*siehe Zeile 285*
367	ESEL,S,ETAB,TABELLE,stress_minimum_%etappe%*1000,s_2_%etappe%, ,0	
368	*GET,e_zahl,elem,0,count	*siehe Zeile 93, entsprechend geht es hier um Elemente*
369	*if,e_zahl,gt,0,then	*siehe Zeile 202*
370	nsle,s,all	*siehe Zeile 50*
371	esln,s,0,all	*siehe Zeile 49*
372	cm,alive,elem	*siehe Zeile 48*
373	/solu	*siehe Zeile 1*

Zeile	Befehl	Erklärung
374	ealive,all	*belebt bestimmte Elemente wieder, wenn sie vorher deaktiviert waren, dann werden sie wiederbelebt und wenn sie schon lebten, dann leben sie weiterhin: ealive KOMMA ALL (All=alle selektierten Elemente werden wiederbelebt, falls man nichts eingibt, wird all als Standard angenommen; will man nur ein bestimmtes Element wiederbeleben, kann man hier ansonsten die entsprechende Elementnummer angeben)*
375	esel,s,live	*siehe Zeile 177*
376	nsel,all	*siehe Zeile 30*
377	lsclear,all	*siehe Zeile 31*
378	LSSOLVE,1,n,1	*siehe Zeile 32*
379	/POST1	*siehe Zeile 33*
380	*do,ls,1,n	*siehe Zeile 34*
381	LCDEF,ls,ls	*siehe Zeile 35*
382	*enddo	*siehe Zeile 36*
383	*do,ls,1,n	*siehe Zeile 37*
384	LCASE,ls	*siehe Zeile 38*
385	*enddo	*siehe Zeile 39*
386	SUMTYPE, PRIN	*siehe Zeile 40*
387	*do,ls,1,n	*siehe Zeile 41*
388	LCOPER,MIN,ls	*siehe Zeile 42*
389	*enddo	*siehe Zeile 43*
390	/title,Etappe %etappe%, Iteration Nr. %iter%, Alive	
391	!*	*siehe Zeile 58*
392	PLNSOL,S,3,0,1.0	*siehe Zeile 59*
393	!*	*siehe Zeile 60*
394	*endif	*bezieht sich auf Zeile 369*
395		
396		
397	esel,s,live	*siehe Zeile 177*
398	etable,tabelle,s,3	*siehe Zeile 285*
399	ESEL,S,ETAB,TABELLE,stress_minimum_%etappe%*1000,stress_grenze_%etappe%, ,0	
400	cmsel,a,alive	*siehe Zeile 160*
401	cmsel,u,elem_no_kill_ref	*siehe Zeile 160*
402	*GET,e_zahl,elem,0,count	*siehe Zeile 93, entsprechend geht es hier um Elemente*
403	*if,e_zahl,gt,0,and,ref_var_%etappe%,le,ref_wieoft_%etappe%,then	
404	*if,mod(iter,ref_freq_%etappe%),eq,0,then	
405	ref_var_%etappe%=ref_var_%etappe%+1	*Namenserklärungen aus Zeile 81 beachten, hier wird ref_var_%etappe% gleich ref_var_%etappe% plus eins gesetzt*

Zeile	Befehl	Erklärung
406	/prep7	*siehe Zeile 17*
407	nsel,all	*siehe Zeile 30*
408	lsclear,all	*siehe Zeile 31*
409	nsle,s,all	*siehe Zeile 30*
410	nref,all,,,,,,off	*macht eine Verfeinerung bei bestimmten Knoten nach bestimmten Kriterien: nref KOMMA all (wenn man alle selektierten Knoten verfeinern will, gibt man hier all ein, ansonsten gibt man hier die Nummer des ersten zu verfeinernden Knoten an, dann nach dem KOMMA die die Nummer des letzten zu verfeinernden Knoten und nach dem KOMMA danach die Inkrementzahl, also die zahl in welchen Schritten die zu verfeinernden Knoten aufeinander folgen) KOMMA KOMMA KOMMA 1_oder_2_oder_3_oder_4_oder_5(wie fein sol die verfeinerung sein? 1=sehr_grob;5=sehr_fein, falls hier nichts angegeben wird, wird 1 angenommen) KOMMA 1_oder_2_oder_3 (wie tief soll die Verfeinerung gehen, also wie weit soll die Umgebung mitverfeinert werden? 1=gar nicht; 2=mehr..., falls hier nichts angegeben wird, wird 1 angenommen) KOMMA off_oder_clean_oder_smooth (Nachbehandlung der neu entstandenen Elemente) KOMMA on_off (on=bei viereckigen Elementen bleiben; off=viereckige Elemente wenn nötig auch in dreiecke spalten)*
411	esel,s,live	*siehe Zeile 177*
412	/SOL	*siehe Zeile 1*
413	nsel,all	*siehe Zeile 30*
414	lsclear,all	*siehe Zeile 31*
415	LSSOLVE,1,n,1	*siehe Zeile 32*
416	/POST1	*siehe Zeile 33*
417	*do,ls,1,n	*siehe Zeile 34*
418	LCDEF,ls,ls	*siehe Zeile 35*
419	*enddo	*siehe Zeile 36*
420	*do,ls,1,n	*siehe Zeile 37*
421	LCASE,ls	*siehe Zeile 38*
422	*enddo	*siehe Zeile 39*
423	SUMTYPE, PRIN	*siehe Zeile 40*
424	*do,ls,1,n	*siehe Zeile 41*
425	LCOPER,MIN,ls	*siehe Zeile 42*
426	*enddo	*siehe Zeile 43*
427	/title,Etappe %etappe%, Iteration Nr. %iter%, Refinement	
428	!*	*siehe Zeile 58*
429	PLNSOL,S,3,0,1.0	*siehe Zeile 59*
430	!*	*siehe Zeile 60*
431		
432	cmsel,s,kps_bc,kp	*siehe Zeile 160*
433	nslk,s	*siehe Zeile 98*

Zeile	Befehl	Erklärung
434	esln,s,0	*siehe Zeile 49*
435	nsle,s	*siehe Zeile 50*
436	cm,Knoten_no_kill_ref,Knoten	*siehe Zeile 48*
437	esln,a,0	*siehe Zeile 49*
438	cm,elem_no_kill_ref,elem	*siehe Zeile 48*
439	*endif	*bezieht sich auf das *if von Zeile 404*
440	*endif	*bezieht sich auf das *if von Zeile 403*
441		
442	/UI,COPY,SAVE,png,GRAPH,COLOR,reverse,portrait,yes	
443	*enddo	*bezieht sich auf das *do von Zeile 222*
444		
445	*enddo	*bezieht sich auf das *do von Zeile 133*

Eidesstattliche Versicherung

Zoghian, Mohamad-Sadegh

Name, Vorname

Ich erkläre hiermit an Eides statt,

dass ich die vorliegende Dissertation mit dem Thema

Entwicklung eines Algorithmus zur Simulation von in-vivo-Knochenveränderungen in FE-Programmen

selbständig verfasst, mich außer der angegebenen keiner weiteren Hilfsmittel bedient und alle Erkenntnisse, die aus dem Schrifttum ganz oder annähernd übernommen sind, als solche kenntlich gemacht und nach ihrer Herkunft unter Bezeichnung der Fundstelle einzeln nachgewiesen habe.

Ich erkläre des Weiteren, dass die hier vorgelegte Dissertation nicht in gleicher oder in ähnlicher Form bei einer anderen Stelle zur Erlangung eines akademischen Grades eingereicht wurde.

Köln, 08.10.2016

Ort, Datum

Unterschrift Doktorandin/Doktorand