Masarykova univerzita
Fakulta informatiky

Pokročilé metody modelování elastických deformací v prostředí MATLAB

Diplomová práce

Bc. Petra Ondřejková

Brno, jaro 2017

Masarykova univerzita
Fakulta informatiky

Pokročilé metody modelování elastických deformací v prostředí MATLAB

Diplomová práce

Bc. Petra Ondřejková

Brno, jaro 2017

Na tomto místě se v tištěné práci nachází oficiální podepsané zadání práce a prohlášení autora školního díla.

Prohlášení

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracovala samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používala nebo z nich čerpala, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Bc. Petra Ondřejková

Vedoucí práce: RNDr. Igor Peterlík, Ph.D.

Poděkování

Na tomto místě bych chtěla poděkovat svému vedoucímu diplomové práce RNDr. Igoru Peterlíkovi, Ph.D. za odborné vedení, cenné rady, trpělivost a ochotu pomoci při zpracování této práce.

Shrnutí

Počítačové simulace měkkých tkání na základě hyperelastických materiálů se v poslední době stává důležitým faktorem ve vzdělání lékařů, předoperačním plánování i při samotné operaci. Matematické a numerické modely jsou postaveny na technikách vyvinutých v oblasti inženýrství, ale pro modelování vysoce deformovatelných měkkých tkáních musí být tyto techniky modifikovány. V současné době existuje několik softwarových prostředí (FeBio, SOFA, atd.), které umožňují simulace měkkých tkání. Protože tyto softwary kladou velký důraz na vysoký výkon, je u nich prakticky vyloučena možnost rychlého prototypování nových metod. Z tohoto důvodu vznikla před mnoha lety matlabovská implementace hyperrealistických simulací.

Práce se zaměřuje na modularizaci existujícího simulačního prostředí, implementaci nové metody pro simulaci kontaktu mezi deformovatelným tělesem a pevnou překážkou. Stávající prostředí disponuje pokročilou metodou rekonstrukce nedeformované geometrie tělesa a tak jsou v práci zkoumány možnosti přidání kontaktů do této metody. Součástí práce je také vytvoření dokumentace s simulačnímu prostředí, aby bylo možné prostředí zveřejnit.

Klíčová slova

Elasticita, metoda konečných prvků, kontakty, numerické metody, simulace v medicíně, Matlab, rekonstrukce geometrie klidové tělesa, problém lineární komplementarity, kompenzace sil

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

První počítačové simulace reálných dějů můžeme datovat do doby vzniku prvních počítačů. V dnešní době se už člověk ani nepozastavuje nad počítačovými simulacemi v technických oblastech, jako jsou stavebnictví (zátěžové testy mostů, atd.), strojírenství (crash testy vozidel, simulace opotřebení součástek různých zařízení, aj.) nebo v letectví, kde jsou používány letecké simulátory pro výcvik pilotů.

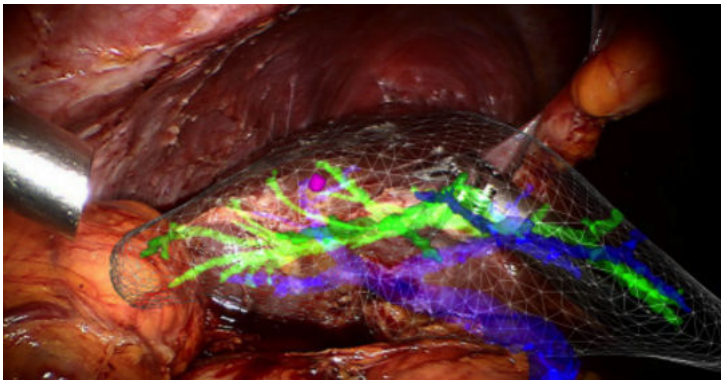
V poslední době pokrok v reálnosti počítačových simulací zapříčinil, že se do této skupiny můžeme zařadit novou oblast – lékařství. Zde počítačová simulace a virtuální realita najdou uplatnění v různých oborech, mimo jiné v chirurgii, kde počítačové simulace nejen snižují náklady, ale také zachraňují lidské životy. V následujících odstavcích si uvedeme pár příkladů.

První využití souvisí s výukou chirurgů. Je až s podivem, že výuka chirurgů, kteří mají v rukou lidské životy, je na míle vzdálená od výukových metod pilotů. Dokážete si představit, že právě vystudovaný pilot po několika pozorovacích letech (kdy pilot pouze pozoruje práci zkušenějšího pilota) bude pilotovat? U pilota je tento scénář nereálný, ten musí nalétat mnoho hodin na leteckém simulátoru a pak je mu povoleno sednout do opravdového letadla. Vytvoření simulátoru operací pomocí virtuální reality a počítačových simulací by mohlo pomoci k zkvalitnění vzdělání chirurgů, kteří by si mohli zcela bez rizika zkoušet a prohlubovat své dovednosti [6]. Tato myšlenka se stává realitou. Na světě existuje již několik simulačních center virtuální chirurgie; jedno takové se nachází i Brně ve Fakultní nemocnici u svaté Anny [34].

Další možnou aplikací počítačových simulací je při plánování operací. Jedná se o vytvoření matematického modelu daného pacienta, na němž by pak operační tým mohl plánovat a zkoušet operační zákrok. Předešlo by se spekulacím o tom, jakou metodu při operaci použít. Navíc by si operační tým mohl díky simulaci postup operace několikrát vyzkoušet a snížit tak riziko chyby. Možná to někoho překvapí, ale dnes už nejsme tak daleko od podobného plánování. Nedávno bylo provedeno několik chirurgických zákroků při jejichž plánování byly využity počítačové modely pacientů, jen zatím místo virtuální reality,

kde by si mohli chirurgové operaci vyzkoušet, byly použity modely pacientů s využitím 3D tisknutí a filmových speciálních efektů [32].

Posledním, ale taky velmi důležitým využitím simulací, je užití rozšířené reality u operací, jež jsou prováděny pomocí haptických zařízení, kdy je chirurg závislý na kameře, která stěžuje orientaci chirurga během operace. Rozšířená realita by mohla chirurgům pomoci představit si, kde přesně se operovaný nádor nachází a jak se k němu dostat bez poškození důležitých cév. V praxi by chirurg během operace měl k dispozici monitor, na který by mu byl promítán reálný obraz i virtuální, jenž by byl založen na matematickém modelu [15, 24].



Obrázek 1.1: Rozšířená realita při operaci jater. Převzato z [14].

Na obrázku 1.1 vidíme rošířenou realitu v praxi. Fotografie je z laparoskopické operace jater, kdy se chirurg snaží odstranit nádor, který se v játrech nachází. Nádor je na obrázku zobrazen růžovou barvou, zelenou a modrou barvou jsou znázorněny cévy jater.

Z předchozích odstavců vyplývá, že v dnešní době jsou simulace velmi důležitým nástrojem, na který jsou kladeny stále větší a větší požadavky z hlediska diverzifikace. Zkuste si modelovat stejnými rovnicemi deformaci mostu a jater. Nelineární chování tkání nemůže být vyjádřeny rovnicemi předpokládající lineární vztahy mezi veličinami. I člověk bez hlubšího matematického vzdělání musí uznat, že matematické a numerické modely používané pro simulace v oblasti inženýrství (variační metody metoda konečných prvků, atd.) nemohou být využity přímo pro modelování tkání. Není ale zapotřebí vymýšlet nové techniky, stačí pouze stávající metody vhodně upravit a rozšířit

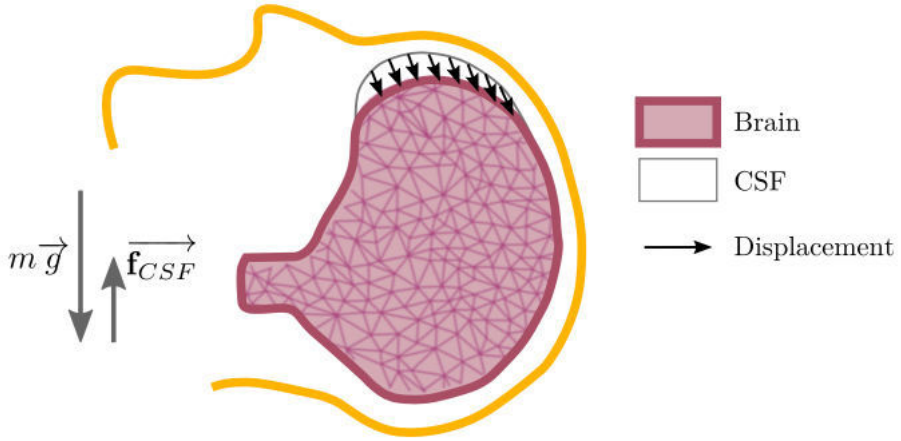
tak, aby zohledňovaly nelineární chování tkání a také jejich interakci s okolím.

V současné době existují programy, které umožňují počítat tyto simulace, např. FeBIO, SOFA, ArtiSynth a další. S jejich pomocí můžeme modelovat působení různých sil a dokonce i kontakty mezi tělesy. Nevýhodou těchto programů je důraz na výkon, který znemožňuje rychlého prototypování nových metod. Z tohoto důvodu vznikla matlabovská implementace hyperelastických simulací (MatlabFEM), která umí pro libovolnou síť složenou s čtyřstěnů spočítat deformace způsobené působícími silami, tlaky či předepsaným posunutím. Matlab jakožto nástroj matematického modelování obsahuje implementace základních matematických funkcí, a tak tyto metody nemusí být při prototypování psát programátor, ale pouze si najde odpovídající funkci Matlabu.

MatlabFEM umí simulovat deformaci tělesa pomocí gravitace, tažných sil, sil aplikovaných ve všech bodech tělesa či pomocí předepsaného posunutí pro jednotlivé body tělesa. Ve své původní verzi program neobsahuje metody modelování kontaktů, a to ani nejjednodušší verzi kontaktů mezi deformovatelným tělesem a pevnou (nedeformovatelnou) překážkou.

Součástí MatlabFEMu je nicméně pokročilá iterační metoda rekonstrukce nedeformované pozice tělesa ze znalosti působících sil (*metoda kompenzace sil*), která může být velmi perspektivní metodou v oblasti medicínských simulací. Modely využívané v simulacích jsou totiž rekonstuovány z medicínských dat, např. z CT či MRI obrazů získaných akvizicí na konkrétním pacientovi. Zobrazované orgány jsou nicméně v době skenování vystaveny působícím silám: pacient se totiž nachází v gravitačním poli země, které působí na orgány, které jsou tudíž permanentně vystaveny mechanickému pnutí. Znalost tohoto pnutí je důležitá pro přesnou simulaci biomechanického chování orgánů, nicméně nelze jí extrahovat z naskenovaných obrazů, které poskytují informace pouze o geometrii orgánů. Metoda implementovaná v prostředí MatlabFEM provádí odhad mechanického pnutí právě pomocí odhadu nedeformované konfigurace. Kromě např. abdominálních orgánů, přímo vystavených gravitační síle, lze využit této metody pro zpřesnění simulací neurochirurgických zákroků. Z důvodu umístění mozku v uzavřené lebce naplněné mozkomíšní tekutinou (*cerebrospinal fluid*, CSF), která je pod tlakem, je mozek neustále v deformované

pozici. Při operaci mozku dochází ale k odstranění částí lebky popř. k navrtání lebečné kosti, což vede k úniku mozkomíšní tekutiny a tudíž ke ztrátě intrakraniálního tlaku. To finálně vede k nezanedbatelné změně geometrických a fyzikálních vlastností mozku (tzv. *brain shift* [4]). Dopředná zalost této změny může mít signifikantní dopad na samotné plánování operace. Na obrázku 1.2 je deformace / posun



Obrázek 1.2: Ukázka deformace mozku díky působení gravitace. Převzato z [4].

mozku na ležícího pacienta způsobené gravitací a tlakem mozkomíšní tekutiny uvnitř lebky.

Cílem této práce bylo nejdříve modularizovat simulační prostředí MatlabFEM a upravit konfiguraci simulace tak, aby bylo jednoduché měnit a upravovat komponenty výpočtu a zkoušet nové techniky řešení. Pak jsme se zaměřili na rozšíření prostředí o novou metodu, která by umožňovala modelovat nejjednodušší variantu kontaktů, a to interakci deformovatelného tělesa s pevnou překážkou; pro jednoduchost detekce kolizí uvažujeme rovinu. Dalším z cílů práce bylo prozkoumat možnosti přidání kontaktů do metody kompenzace sil. A v neposlední řadě měla v rámci práce vzniknout anglická dokumentace prostředí MatlabFEM, aby bylo možné celý program vystavit veřejně.

2 Modelování deformací

V této kapitole nahlédneme do světa matematického modelování deformací těles. Definujeme zde základní pojmy používané při modelování pohybujících se těles. Ukážeme si, že existují dva způsoby popisu pohybu, jejich rozdílnost si uvedeme na pár zajímavých příkladech. Neopomeneme uvést několik pojmů z teorie elasticity, která je jednou ze základních oblastí matematického modelování pohybu těles. Vysvětlíme si pojmy mechanické napětí (*stress*), deformace (*strain*) a uvedeme tzv. konstituční rovnici. Stručně vysvětlíme metodu konečných prvků, jež nám pomáhá převést nekonečný počet stupňů volnosti, se kterým teorie elasticity pracuje, do konečné roviny uchopitelné výpočetní technikou. A na závěr si ukážeme, jak se soustavy rovnic vzniklé z uvedených definic řeší.

Pro lepší orientaci čtenáře v textu zavedeme matematické značení, které budeme využívat v celé této části práce.

Vektory jakožto ukazatele velikosti a směru budeme v textu značit tučnými malými písmeny např. $\mathbf{v}, \mathbf{w}, \mathbf{f}$, abychom je odlišili od obyčejných skalárů, které budeme značit malými písmeny abecedy např. a, b, c. Tensory [1], které jsou pouze zobecnění pojmu vektor, budeme značit malými písmeny řecké abecedy např. σ, δ. Matice budeme v textu značit velkými tečnými písmeny abecedy např. $\mathbf{E}, \mathbf{S}$., prvky matice pak příslušnými malými písmeny abecedy s indexem např. e_{ij} značí prvek matice $\mathbf{E}$ na i řádku a j sloupci. Jednotkovou matici budeme značit $\mathbf{I}$.

2.1 Eulerovský vs. Lagrangeovský pohled na pohyb a deformaci tělesa

V mechanice kontinua můžeme na pohyb a deformaci tělesa pohlížet ze dvou stran a popisovat je tak dvěma způsoby – Eulerovským popř. Lagrangeovským přístupem. Pojďme si nyní oba způsoby vysvětlit.

---

1. Tensory 1. řádu jsou vektory. Tensory 2. řádu jsou matice, které jsou symetrické podle diagonály. A zajímavé je také dodat, že tensory 2. řádu, velikosti 3 (vlastně matice 3x3), definují elipsoid: vlastní čísla tensoru jsou souřadnice určují souřadnice ohniska a vlastní vektory pak orientaci os.

- **Eulerovský pohled:** popisuje mechaniku tělesa na základě znalosti mechaniky tělesa v pevně daném „okně“, přes které na těleso pohlíží. Jako kdybychom seděli doma na židli a skrz okno jsme se snažili popsat pohyb vlaku, který kolem okna projíždí.
- **Lagrangeovský pohled:** popisuje mechaniku tělesa na základě mechaniky pevně zvolené částice popř. skupiny částic. Vrátíme-li se k našemu vlaku, znamenalo by to, že bychom se snažili popsat pohyb vlaku na základě pozorování pohybu např. lokomotivy, kterou sledujeme po celou dobu jejího pohybu z vrtulníku letícího nad vlakem.

Pro lepší pochopení rozdílnosti těchto pohledů si uveďme příklad. Uvažujme vertikálně pruhované těleso (válec, kvádr, atd.), které natahujeme v horizontálním směru. Nyní se budeme snažit popsat tento pohyb tělesa na základě pozorování barvy tělesa.

Eulerovským pohledem pozorujeme těleso jakoby přes „okno“. Jak těleso natahujeme, vidíme pohybující se jednotlivé barevné pruhy, což znamená, že funkce, která bude popisovat barvu tohoto tělesa, bude funkcí času.

Lagrangeovský pohled si na tělese vybere jednu částici a tu bude sledovat po celou dobu pohybu. Jelikož se barva pohybující se částice nemění, funkce popisující změnu barvy tělesa v čase bude konstanta.

Na příkladu je pěkně vidět, že oba přístupy popisují pohyb jinak a jsou tak vhodné pro popis mechaniky těles různých vlastností. Např. Lagrangeovský pohled není vhodný pro popis mechaniky kapalin a plynů, protože zde obvykle nejsme schopni sledovat jednotlivé částice po celou dobu pohybu, ale naopak pozorujeme objem, přes který částice proudí [3].

2.1.1 Systémy souřadnic

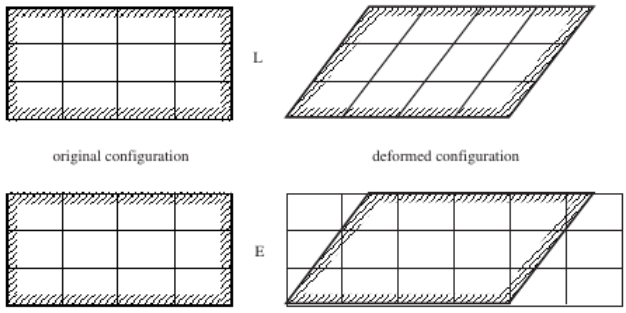
Systémy souřadnic hrají velkou úlohu při popisu deformace a pohybu tělesa. Díky existenci dvou pohledů na deformaci/pohyb tělesa, vznikly také dva rozdílné pohledy na tyto systémy, které se dají mezi sebou převádět pomocí tzv. deformačního gradientu (více a deformačním gradientu v kapitole 2.3.2). Jedná se o:

- **Prostorový souřadný systém:** klasický kartézský systém souřadnic, do něhož umístíme těleso, které deformujeme. V průběhu

deformace se souřadný systém vůči okolnímu světu nijak nemění. V souřadném systému, ale dochází k deformaci/pohybu tělesa, tzn. těleso mění svou polohu vzhledem k počátku souřadného systému.

- **Materiálový souřadný systém:** souřadný systém umístěný na deformujícím se tělese, který se v průběhu deformace mění tak, jak se deformuje těleso. Pohybující/deformující se těleso nemění svou polohu vzhledem k počátku tohoto souřadného systému.

Z popisu obou systémů plyne, že prostorový souřadný systém odpovídá Eulerovskému pohledu a materiálový souřadný systém Lagrangeovskému pohledu [27].



Obrázek 2.1: Rozdíl mezi prostorovým a materiálovým souřadným systémem. Převzato z [3].

Obrázek 2.1 zachycuje rozdílné chování prostorového a materiálového souřadného systému při deformaci. Z obrázku je patrné, že prostorový souřadný systém se deformací nemění (dolní dva obrázky) a materiálový souřadný systém se deformuje spolu s tělesem (horní dva obrázky).

2.2 Konfigurace tělesa

Představme si nyní, že máme deformující se těleso např. kvádr, který taháme silou do jednoho směru. Tuto deformaci jsme nasnímali fotoaparátem. Fotografie tělesa můžeme rozdělit do 2 základních skupin na základě toho, jakou konfiguraci tělesa ztělesňují.

- **Základní konfigurace:** Vyjadřuje stav tělesa, před tím než na něj začala působit síla, která těleso deformuje. V případě fotografií by se jednalo o fotografie pořízené před deformací kvádru.
- **Konečná konfigurace:** Vyjadřuje stav tělesa, po nastolení rovnováhy vnějších deformačních sil (síly, které na těleso působí z venku a deformují ho) a vnitřních sil (působících proti deformační síle). K nastolení rovnováhy (ekvilibria) dochází v případě, že součet působení obou těchto sil je nulový a těleso tak zůstává v klidu. V případě fotografií se jedná o fotografie, kdy na těleso stále působí síla, ale těleso už se nepohybuje popř. nedeformuje.

Pro jednodušší popis rozdílů mezi rozdílnými výpočty mechanického napětí si zaveďme ještě pojem **referenční konfigurace**. Referenční konfigurací označme konfiguraci tělesa, vůči které vyjadřujeme síly v daném kroku výpočtu [17].

2.3 Elasticita

V reálném světe jsme obklopeni pružnými tělesy. Každé takové těleso lze do jisté míry deformovat tzn. měnit jeho tvar, objem nebo rozměry působením vnějších sil. A tak musí existovat část fyziky, která popisuje tuto deformaci a síly, které při ní působí. Tou oblastí je teorie elasticity, která definuje pojmy jako napětí, deformace, deformační gradient, atd. a na základě nich se snaží deformaci těles popsat.

2.3.1 Druhy deformace

Každé pevné těleso v reálném světě je lze působením sil deformovat. Jako příklad takového tělesa si uveďme pružinu, kterou můžeme působením síly natáhnout, nebo železnou tyč, kterou může dostatečně velká síla ohnout. Z rozdílného chování těles při deformaci rozlišujeme 2 druhy deformace.

- **Pružná (elastická):** Pokud přestaneme na těleso působit vnějšími silami, těleso se vrátí do stavu před začátkem působení vnějších sil (např. pružina, která se pokud na ni přestaneme působit silou vrátí zpátky).

- **Lineární elasticita:** Vztah mezi mechanickým napětím a deformací je lineární, platí Hookův zákon, který říká, že deformace je přímo úměrná napětí. Tato oblast postihuje jen velmi malé procento deformací a to takové, při nichž působí malé síly.

- **Hyperelasticita:** Vztah mezi mechanickým napětím a deformací není lineární, nemůžeme zde uplatnit znalost Hookova zákona. Do této kategorie patří i modelování měkkých tkání. Dochází zde po aplikaci sil k velkým tvarovým změnám.

- **Hypoelasticita:** Rozšíření varianty lineárního modelu.

- **Tvárná (plastická):** Pokud přestaneme na těleso působit vnějšími silami, těleso zůstane ve zdeformované stavu (např. ohnutá železná tyč, na kterou když přestaneme působit, tak se zpátky nenarovná)

Deformace každého tělesa prochází svým vývojem, který můžeme popsat pomocí napětí / tlaku. To je definováno jako:

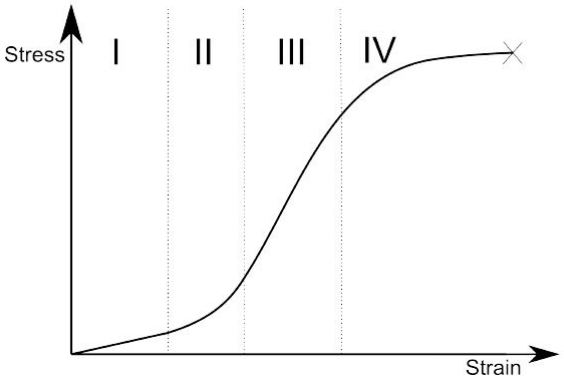
$$\sigma = \frac{F}{S}[Pa] \tag{2.1}$$

kde F je velikost síly působící kolmo na plochu příčného řezu o obsahu S. Při vývoji deformace elastické deformace přechází do plastické, pokud napětí tělesa překročí určitou mez označovanou jako mez pružnosti σ_E. Ta je pro každé těleso jiná a její hodnota se dá experimentálně změřit. Pokud by i po překročení meze pružnosti docházelo k zvětšování působící síly a hodnota napětí by dále rostla. Mohla by tak překročit další významnou mez označovanou jako mez pevnosti σ_P a došlo by k porušení struktury tělesa. Inženýři umí tento průběh deformace tělesa zaznamenat do grafu, která pak nazývají křivkou deformace (obrázek 2.2). [13]

Z křivky deformace pak můžeme pozorovat, že v oblasti lineární elasticity je napětí σ přímo-úměrné relativní deformaci tělesa (konstanta úměrnosti se nazývá Youngův modul E), tzn. že dostáváme rovnost (Hookův zákon):

$$\frac{F}{S} = E\frac{\Delta d}{d}, \tag{2.2}$$

kde d je původní délka tělesa a Δd je změna délky tělesa při aplikaci síly.



Obrázek 2.2: Křivka deformace pro lidskou tkáň. Část (I) je oblastí lineární elasticity. S růstem deformace dochází v oblasti (II) k narovnávání kolagenových vláken, oblast (III) představuje část, kdy jsou vlákna narovnaná a křivka se stává lineární a v oblasti (IV) dochází k plastické deformaci tkáně s následným přetržením. Převzato z [18]

V této práci se omezíme pouze na elastickou deformaci, ta tvoří základ matematického modelování deformací ve všech oborech. A teorie elasticity mám dává do ruky nástroje, kterými můžeme jednoduše popsat deformaci pružných těles. Celá teorie elasticity spojuje dva jednoduché principy popisu chování tělesa na kinetickou a kinematickou část [13].

- **Kinematika:** Popisuje pohyb pomocí vzdálenost, rychlosti nebo zrychlení, nezajímá jí zdroj pohybu. V teorii elasticity se využívá jako měřidlo deformace (viz. kapitola 2.3.3).

- **Kinetika:** Popisuje pohyb v závislosti na tom, co ho způsobilo (např. tlak, vnitřní a vnější síly atd.). Jako měřidlo se v teorii elasticity využívá mechanické napětí (viz. kapitola 2.3.4).

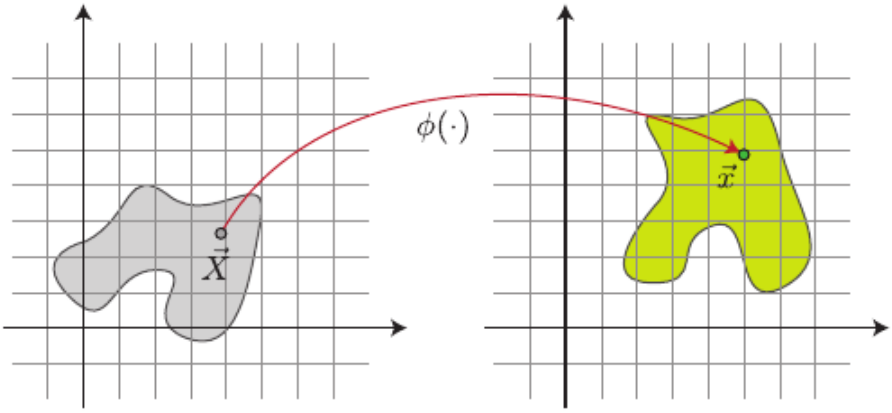
2.3.2 Mapování deformace a deformační gradient

Deformaci elastického tělesa si můžeme představit jako funkci, která popisuje vztah mezi počátečním stavem a deformovaným stavem tělesa. Graficky znázorněno na obrázku 2.3. Tato funkce se většinou

nazývá deformační funkcí $\boldsymbol{\phi}$ a ve 3D ($\boldsymbol{\phi} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$) ji můžeme zapsat:

$$\mathbf{x} = \boldsymbol{\phi}(\mathbf{X}) \tag{2.3}$$

kde $\mathbf{X}, \mathbf{X} \subset \mathbb{R}^3$ označuje základní nebo referenční konfiguraci tělesa[2] a $\mathbf{x}, \mathbf{x} \subset \mathbb{R}^3$ označuje konečnou konfiguraci tělesa po deformaci.



Obrázek 2.3: Deformační funkce. Převzato z [29].

Z deformační funkce můžeme vypočítat deformační gradient, který je Jakobiánem (směrové parciální derivace x,y,z) této funkce. Ten nám říká, jak moc se souřadnice daného bodu mění deformací ve všech směrech (x,y,z). Matematicky můžeme deformační gradient pro těleso v 3D souřadném systému napsat jako:

$$F = \frac{\partial\left(\phi_1, \phi_2, \phi_3\right)}{\partial\left(X_1, X_2, X_3\right)} = \begin{pmatrix} \frac{\partial \phi_1}{\partial X} & \frac{\partial \phi_1}{\partial Y} & \frac{\partial \phi_1}{\partial Z} \\ \frac{\partial \phi_2}{\partial X} & \frac{\partial \phi_2}{\partial Y} & \frac{\partial \phi_2}{\partial Z} \\ \frac{\partial \phi_3}{\partial X} & \frac{\partial \phi_3}{\partial Y} & \frac{\partial \phi_3}{\partial Z} \end{pmatrix} \tag{2.4}$$

kde $\mathbf{X} = (X, Y, Z)^T$ představuje těleso a $\boldsymbol{\phi}(\mathbf{X}) = (\phi_1(\mathbf{X}), \phi_2(\mathbf{X}), \phi_3(\mathbf{X}))^T$ je deformační funkce. Takový zápis funkce znamená, že deformační funkci můžeme napsat jako vektor funkcí x, y, z [29].

2. Těleso je teorii elasticity považováno jako spojitý objekt, složený z nekonečného počtu částic. Jeho popis tak vyžaduje použití spojitých funkci a analytických řešení, což ale obvykle není možné. Proto se při reálném výpočtu používá diskretizace.

Pro lepší představu si můžeme uvést pár příkladů deformační funkce a deformačního gradientu:

- TRANSLACE
  Deformační funkci pro posunutí neboli translaci není tak těžké vymyslet. Pokud uvážíme množinu bodů, které nám reprezentují nějaké těleso, tak posunutí tohoto tělesa můžeme realizovat jako přičtení konstantní hodnoty ke každé souřadnici jednotlivých bodů. To matematicky zapíšeme jako:

  $$\boldsymbol{\phi}(\mathbf{X}) = (X + a, Y + b, Z + c)^T, \tag{2.5}$$

  kde a, b, c jsou konstanty, o které chceme těleso v daných souřadnicích posunout. Deformačním gradientem je směrové derivace a tou je jednotková matice $\mathbf{I}$.

- ZVĚTŠENÍ/ZMENŠENÍ
  Deformační funkci pro škálování není také problém jednoduše odvodit. Pokud uvážíme množinu bodů, které nám reprezentují nějaké těleso, tak škálování provedeme jako vynásobení všech souřadnic bodů stejnou konstantní hodnotou. Matematicky zapíšeme jako:

  $$\boldsymbol{\phi}(\mathbf{X}) = (aX, aY, aZ)^T, \tag{2.6}$$

  kde a je konstantní hodnota určující, zda se jedná o zvětšení ($a > 1$), zmenšení ($a \in (0, 1)$) nebo identitu ($a = 1$). Pokud si tedy k translaci spočítáme deformační gradient zjistíme, že tím je a násobek jednotkové matice $a\mathbf{I}$.

- ROTACE
  S deformační funkcí pro rotaci tělesa ve 3D to už není tak lehké, dá se dohledat v literatuře. V [5] najdeme, že rotaci tělesa kolem osy X o úhel α můžeme zapsat následovně:

  $$\boldsymbol{\phi}(\mathbf{X}) = (X, Y\cos\alpha + Z\sin\alpha, -Y\sin\alpha + Z\cos\alpha)^T, \tag{2.7}$$

  Po spočítání deformačního gradientu dostáváme pro většinu lidí známější variantu matice rotace, kterou stačí bod, který chceme

rotovat vynásobit a výsledný vektor jsou souřadnice rotovaného bodu.

$$\mathbf{F} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{pmatrix} \tag{2.8}$$

Navíc u matice rotace je $\mathbf{F}$ vlastní rotační maticí a platí tak následující vztah mezi maticí rotace a její transpozicí: $\mathbf{F}^\top\mathbf{F} = \mathbf{I}$. [29]

2.3.3 Deformace (*strain*)

S deformací tělesa úzce souvisí změna (zvýšení) potenciálové energie tělesa (práce, která je vykonána, aby došlo k deformaci tělesa), která je v mechanice kontinua úzce spojena s pojmem deformační energie (*strain energy*) $E[\phi]$. Jak je vidět z označení, funkce úzce souvisí s deformační funkcí ϕ. U elastických materiálů platí, že velikost potencionální energie deformovaného tělesa závisí pouze na základní a konečné konfiguraci tělesa, z čehož plyne, že funkce není závislá na cestě, jakou byla deformace dosažena.

V mechanice kontinua je deformační energie definována jako:

$$E[\phi] = \int_\Omega \Phi[\phi, \mathbf{X}]d\mathbf{X}, \tag{2.9}$$

kde Φ je hustota energie (*enegy density*), která je funkcí deformační funkce ϕ a základní konfigurace tělesa $\mathbf{X}$. Hustota energie vyjadřuje energii deformace na jednotku nedeformovaného objemu tělesa přes všechny body tělesa.

Navíc lze ukázat, že funkci $\Phi[\phi, \mathbf{X}]$ lze zapsat jako funkci deformačního gradientu:

$$\Phi[\phi, \mathbf{X}] = \Phi[\mathbf{F}(\mathbf{X})] = \Phi(\mathbf{F}), \tag{2.10}$$

kde Φ je hustota energie a $\mathbf{F}(\mathbf{X})$ je lokálního deformační gradient.

Jako příklad pro lepší pochopení, jak může taková hustota energie vypadat, si můžeme uvést v inženýrství asi nejpoužívanější model pro deformaci a to Greenovu deformaci (*Green strain*):

$$\Phi(\mathbf{F}) = E = \frac{1}{2}(\mathbf{F}^T \cdot \mathbf{F} - I). \tag{2.11}$$

Tento model popisuje deformaci tělesa z hlediska relativního posunutí (*displacement*) každého bodu tělesa. Pokud je deformační gradient $\mathbf{F}$ roven jednotkové matici $\mathbf{I}$, je hodnota Greenovy deformace 0. Pokud je pohyb tělesa roven rotaci popř. translaci, pak $\mathbf{R}^T \cdot \mathbf{R} = \mathbf{I}$ a hodnota Greenovy deformace je opět 0.

Rozepíšeme-li si rovnici pro Greenovu deformaci tak, že deformační gradient nahradíme jeho jinou definicí přes posunutí $\mathbf{F} = \mathbf{I} + \frac{\partial \mathbf{u}}{\partial \mathbf{X}}$ dostáváme [29]:

$$\Phi(\mathbf{F}) = E_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i} + \frac{\partial u_k}{\partial X_i}\frac{\partial u_k}{\partial X_j}\right). \tag{2.12}$$

Z tohoto vztahu vyplývá geometrická nelinearita Greenovy deformace díky poslednímu výrazu $\frac{\partial u_k}{\partial X_i}\frac{\partial u_k}{\partial X_j}$, který je kvadratický.

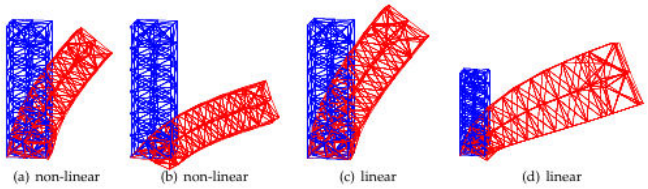
Z důvodu zjednodušení výpočtů bývá v některých částech teorie elasticity tato nelineární forma Greenovy deformace nahrazena lineární variantou, která neuvažuje poslední nelineární člen. Proto můžete v mnoha matematických modelech elastických těles najít tuto definici Greenovy deformace [27]

$$\Phi(\mathbf{F}) = E_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i}\right). \tag{2.13}$$

Na obrázku 2.4 můžeme vidět porovnání výsledků matematických simulací při použitím lineární nebo nelineární definice Greenovy deformace. První dva obrázky zachycují chování chování modelu využívající nelineární definici Greenovy deformace pod malým působením sil (obrázek a) a větším působení sil (obrázek b). Druhé dva obrázky ukazují chování modelu využívající lineární definici Greenovy deformace opět s působením malých (obrázek c) a větších sil (obrázek d). Z obrázků je patrné, že použití lineární definice deformace vede při větších působících silách k nefyzikálnímu narůstání objemu tělesa.

2.3.4 Mechanické napětí (*stress*)

Mechanické napětí je ukazatel deformace tělesa vyjadřující tlak, jaký je na těleso vyvíjen, působíme-li na něj silou popř. tažnou silou. Většinou se zadává pomocí tensorů příslušného řádu (v 3D tensory 3. řádu),



Obrázek 2.4: Porovnání výsledků simulací s užitím lineární a nelineární definicí Greenovy deformace. Převzato z [25].

který nám ukazuje rozklad působící síly do normálových vektorů všech směrů (v 3D x, y, z). Protože mechanické napětí lze definovat více způsoby (volbou pohledu na soustavu souřadnic, ve které tělesa deformujeme), existují 3 základní druhy mechanického napětí a to Cauchyho, 1. Piola-Kirchhoffův a 2. Piola-Kirchhoffův tensor mechanického napětí [3]. Všechny základní druhy mechanického napětí lze mezi sebou převádět.

CAUCHYHO TENSOR NAPĚTÍ (*Cauchy stress*)

Cauchyho tensor napětí nebo také pravé mechanické napětí vychází z Eulerovského pohledu na deformaci tělesa. Fakticky udává velikost síly působící na těleso v normále. Důležité je ale podotknout, že Cauchyho tensor je vyjádřen v materiálových, tudíž zdeformovaných souřadnicích. Tažné síly $\mathbf{t}$ se pomocí něj vyjadřují jako:

$$\mathbf{t} = \sigma \cdot \mathbf{n} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{pmatrix} \cdot \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} \tag{2.14}$$

kde σ je Cauchyho tensor napětí a $\mathbf{n}$ je normálový vektor pro danou tažnou sílu.

3. V další části textu (vyjma nadpisů) budeme 1. a 2. Piola-Kirchhoffův tensor mechanického napětí označovat 1PK nebo 2PK tensor napětí

2. PIOLA-KIRCHHOFFŮV TENSOR NAPĚTÍ (*2PK stress*)

Tato definice mechanického napětí používá v porovnání s Cauchyho tensorem napětí „opačný“ pohled na deformaci tzv. Lagrangeovský pohled. Toto mechanické napětí je vyjadřováno v nedeformovaných souřadnicích na nedeformovaném tělese (tzn. v základním stavu) [17, 27].

1. PIOLA-KIRCHHOFFŮV TENSOR NAPĚTÍ (*1PK stress*)

Je hybridem mezi Cauchyho a 2PK tensorem napětí. Síly reprezentuje pomocí Eulerovského přístupu a deformované těleso (*surface*) zobrazuje pomocí Lagrangianského přístupu. Prakticky se ale při modelování nepoužívá [29].

Jak už bylo zmíněno jednotlivé druhy mechanického napětí se dají mezi sebou převádět. Pro převod Cauchyho a 2PK tensoru napětí slouží následující tzv. Piolův vztah:

$$\sigma = \frac{1}{|\mathbf{F}|}\mathbf{F}\mathbf{S}\mathbf{F}^T \tag{2.15}$$

kde σ označuje Cauchyho tensor napětí, $\mathbf{S}$ je 2PK tensor napětí a $\mathbf{F}$ je deformační gradient.

S pomocí mechanického napětí můžeme vyjádřit statické ekvilibrium celého systému a položit tak základ k řešení celého systému. Nejprve si musíme uvědomit, kdy k ustálení celého systému dochází. Je to tehdy, když jsou síly působící na těleso a vnitřní síly tělesa v rovnováze. Nyní si stačí uvědomit, že vnitřní síly tělesa jsou vyjádřeny pomocí mechanického napětí zapisovaného pomocí tensorů (Cauchy nebo 2PK) a vnější síly jsou pouze matematickým zápisem sil působících na těleso. Tento fakt můžeme zapsat do rovnice:

$$0 = f(\mathbf{x}) + \nabla \cdot \mathbf{s}, \mathbf{x} \in \Omega \tag{2.16}$$

kde f jsou vnější síly působící na těleso, $\mathbf{s}$ je definovaný tensor mechanického napětí.

Ze znalosti této rovnice a dosazením vhodně definovaného mechanického napětí (Cauchy, 1PK, 2PK) můžeme definovat rovnici ekvilibria pro systémy používající různé souřadné systémy. Tak můžeme

vyjádřit rovnováhu v Eulerovském systému (vnější síly $\tilde{f}$ a geometrie tělesa vyjádřeny v deformovaných materiálových souřadnicích) jako:

$$0 = \tilde{f}(\mathbf{x}) + \nabla \cdot \sigma(\mathbf{x}), \mathbf{x} \in \Omega \tag{2.17}$$

kde σ je Cauchyho tensor napětí.

Pokud bychom uvažovali Lagrangeovský pohled (vnější síly $\bar{f}$ a geometrie tělesa vyjádřeny v nezdeformovaných prostorových souřadnicích), pak

$$0 = \bar{f}(\mathbf{x}) + \nabla \cdot [\mathbf{FS}], \mathbf{x} \in \Omega \tag{2.18}$$

kde **S** je 2PK tensor napětí a **F** je deformační gradient.

Na tomto místě si je důležité uvědomit, že pro reálné řešení problému elasticity nelze Eulerovský přístup použít, jelikož vyjádřuje veličiny ve zdeformované konfiguraci, kterou teprve hledáme na základě působících sil. Jelikož je ale řešení problému nelineární elasticity iterativní, existuje analogie Eulerovského pohledu, tzv. *updated Lagrangian*: v tomto případě se fyzikální veličiny vyjadřují v deformovaných souřadnicích, které byly spočteny v předchozí iteraci řešení. Tento přístup pak pro vyjádření rovnováhy používá rovnici 2.17. Klasický Lagrangeovský přístup (veličiny vyjádřeny vždy v nezdeformovaném souřadném systému, 2PK tensor napětí, rovnice 2.18) se pak nazývá *total Lagrangian*. Více o iterativním řešení v sekci 2.5.

2.3.5 Konstituční rovnice

Konstituční rovnice nám ukazuje vztah mezi mechanickým napětím a deformací. Tato rovnice spolu s definicí hustoty energie jsou nejčastější varianty matematické definice vlastnosti materiálu. Jelikož mechanické napětí i deformace jsou uváděny jako symetrické matice 3x3, můžeme vztah mezi oběma vyjadřovat jako vztah mezi původními maticemi nebo použít vektory s délkou 6, které budou obsahovat pouze prvky diagonály a prvky nad touto diagonálou (tzv. *Voigtova notace*). Jak taková konstituční rovnice pro tyto vektory může vypadat

je zde:

$$\begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{pmatrix} = \begin{pmatrix} s_1^0 \\ s_2^0 \\ s_3^0 \\ s_4^0 \\ s_5^0 \\ s_6^0 \end{pmatrix} + \begin{pmatrix} E_{11} & E_{12} & E_{13} & E_{14} & E_{15} & E_{16} \\ E_{21} & E_{22} & E_{23} & E_{24} & E_{25} & E_{26} \\ E_{31} & E_{32} & E_{33} & E_{34} & E_{35} & E_{36} \\ E_{41} & E_{42} & E_{43} & E_{44} & E_{45} & E_{46} \\ E_{51} & E_{52} & E_{53} & E_{54} & E_{55} & E_{56} \\ E_{61} & E_{62} & E_{63} & E_{64} & E_{65} & E_{66} \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \end{pmatrix} \tag{2.19}$$

kde **e** označuje deformační vektor (např. Greenův), **s** vektor mechanického napětí (např. Cauchyho, 1PK nebo 2PK) a $\mathbf{s}^0$ je vektor mechanického napětí v referenční konfiguraci někdy také nazývaný iniciální.

Pokud matice **E** není singulární (neobsahuje lineárně závislé řádky a lze ji spočítat inverze) můžeme vztah mezi mechanickým napětím a deformací vyjádřit následující rovnicí

$$\mathbf{e} = \mathbf{E}^{-1} \left(\mathbf{s} - \mathbf{s}_0 \right) \tag{2.20}$$

Jelikož matice **E** udává vztah mezi vektory délky šest musí obsahovat 36 prvků. (Pokud bychom chtěli matici napsat pro původní matice 3x3, tak by **E** musela obsahovat 81 prvků). Ale jelikož je matice **E** i v obecném případě symetrická a její inverze také, tak je počet koeficientů, které určují vztah mezi mechanickým napětím a deformací pouze 21. Při uvážení isotropních materiálů se počet koeficientů zredukuje pouze na dva, tzv. Lamého koeficienty a celý vztah můžeme přepsat jako:

$$\mathbf{s} = \lambda tr \left(\mathbf{e} \right) \mathbf{I} + 2\mu \mathbf{e} \tag{2.21}$$

kde **s** je mechanické napětí, **e** je deformace, **I** označuje jednotkovou matici, λ je první Lamého koeficient a μ je modul pružnosti ve smyku. Oba tyto koeficienty můžeme snadno dopočítat ze znalosti Youngova modulu (E) a Poissonovy konstanty materiálu (ν) [27].

$$\lambda = \frac{E}{(1+\nu)(1-2\nu)} \qquad \mu = \frac{E}{2+2\nu} \tag{2.22}$$

V oblasti modelování měkkých tkání, se využívají definice materiálů jako jsou St. Venant-Kirchhoff nebo Mooney-Rivlin. Jak jsou tyto materiály definovány (popis konstituční rovnice a hustoty energie) najdete v příloze A.

2.3.6 Formulace problému elasticity

Dáme-li dohromady rovnice vycházející z předchozích vztahů, dostaneme parciální diferenciální rovnici podobnou 2.17. Tato rovnice může být lineární, pokud použijeme lineární variantu tensoru deformace s Hookovým zákonem. Ve většině případů je, ale takto vytvořená rovnice nelineární, nelinearita může být způsobena nelineární definicí Greenova tensoru deformace nebo nelinearitou vztahu mezi deformací a mechanickým napětím popř. obojím.

Abychom byli schopni najít řešení nově vytvořené rovnice, musíme definovat okrajové podmínky. Ty můžeme jednoduše definovat jako posunutí, stačí jen k systému přidat rovnici

$$\mathbf{u}(\mathbf{x}) = \overline{\mathbf{u}}, \mathbf{u} \in \partial\Omega, \tag{2.23}$$

kde $\partial\Omega$ značí část tělesa, kterému posunutí definujeme a $\overline{u}$ je předepsaný posunutí. Takto definované podmínky se nazývají Dirichletovy okrajové podmínky, které předepisují hodnotu primární proměnné a to posunutí.

Kromě těchto okrajových podmínek existují ještě okrajové podmínky, které předepisují hodnotu sekundární veličiny a to tažné síly působící na těleso, které se dají vyjádřit pomocí tensoru napětí (viz. rovnici 2.14).

Výsledný systém, který se snažíme vyřešit po přidání Dirichletových podmínek vypadá takto:

$$0 = f(\mathbf{x}) + \nabla \cdot \sigma(\mathbf{x}), \mathbf{x} \in \Omega \tag{2.24}$$

$$\mathbf{x}(\mathbf{u}) = \overline{\mathbf{u}}, \mathbf{u} \in \partial\Gamma_D \tag{2.25}$$

kde $\partial\Gamma_D$ je část povrch tělesa, na které aplikujeme okrajové podmínky a platí $\partial\Gamma_D \in \partial\Omega$, že , $\overline{\mathbf{u}}$ je hodnota posunutí (nejčastěji je používaná 0, což znamená, že daná část tělesa se nepohybuje). [25]

2.4 Metoda konečných prvků

Výsledkem aplikování rovnic z předchozích kapitol zjišťujeme, že popsat deformace / pohyb tělesa znamená vyřešit soustavu rovnic 2.24 a 2.25. Protože tato soustava ve většině případů nejde vyřešit analyticky, používá se k řešení metoda konečných prvků. Podstatou

metody konečných prvků (MKP, *finite element method, FEM*) je nahrazení spojitého pohledu na těleso, který používá teorie elasticity, za diskrétní.

Postup MKP můžeme rozdělit na tyto základní kroky:

- **Diskretizace Ω:**
  Doména spojitého tělesa je nahrazena množinou prvků nejčastěji trojúhelníků (ve 2D) nebo čtyřstěnů (v 3D). Prvky pokrývají celou doménu tělesa.

- **Konstrukce tvarových funkcí (*shape function*) pro každý element:**
  Tyto funkce jsou interpolační funkce jednotlivých prvků a jsou nenulové pro celou doménu elementu.

- **Převedení formulace na tzv. slabou formu:**
  Integrace pře doménu a násobení testovacími funkcemi.

- **Sestavení celkové matice pružnosti a vektoru sil:**
  Každý prvek pak můžeme zakódovat jako soustavu rovnic, kde jako proměnné vystupují vrcholy jednotlivých elementů. Tuto soustavu můžeme převést do matice a z takto vytvořených matic můžeme poskládat celkovou matici pružnosti $\mathbf{K}$. Stejným způsobem pak poskládáme vektor sil pro jednotlivé elementy.

- **Vyřešení soustavy**

Cílem táto práce není přesné vysvětlení MKP a tak předchozí řádky pouze stručně vysvětlují její podstatu. Detailní popis MKP je možné najít v [19, 30, 33].

Aplikací MKP převedeme spojitý systém 2.24 na soustavu nelineárních rovnic, kterou můžeme zapsat jako:

$$\mathbf{K}(\mathbf{u}) = \mathbf{f} \tag{2.26}$$

kde $\mathbf{K}$ je matice pružnosti, $\mathbf{u}$ jsou hledané pozice vrcholů a $\mathbf{f}$ je vektor působících sil pro jednotlivé vrcholy.

2.4.1 Okrajové podmínky a MKP

Tato část práce popisuje aplikaci okrajových podmínek do systému, na nějž je aplikována metoda konečných prvků. Zaměříme se pouze na případ, kdy aplikujeme Dirichletovy okrajové podmínky (s posunutím), které jsou definovány v kapitole 2.3.6.

Tyto podmínky přidáváme do systému $\mathbf{K}(\mathbf{u}) = \mathbf{f}$, abychom se zbavili singularity matice $\mathbf{K}$. Důležité je uvědomit si, že okrajové podmínky jsou aplikovány na linearizovaný systém, který vznikne při řešení soustavy např. pomocí Newtonovy metody, více o řešení v kapitole 2.5. Existují tři základní možnosti, jak promítnout okrajové podmínky do tohoto systému, a to:

- **Eliminace:** Nejjednodušší variantou aplikace okrajových podmínek. Tato metoda spočívá v tom, že v matici, která znázorňuje systém, vynuluje i řádek a j sloupec, který odpovídá danému vrcholu, jen $\mathbf{K}_{ij} = 1$ a $\mathbf{f}_j = \overline{u}$. Tato metoda je velice jednoduchá, ale jen málo okrajových podmínek jde touto technikou vyjádřit. Mezi podmínky, které je možné tímto způsobem zadat jsou podmínky typu $\mathbf{u}(\mathbf{x}) = \overline{u}$, kde $\overline{u} \in R$.

- **Penalizace:** Tato technika spočívá v nahrazení řádku, jenž odpovídá danému vrcholu, kterého se okrajová podmínka týká, výrazem, kterým definujeme penalizaci

$$\frac{1}{\varepsilon}\left(\mathbf{Q}^T\mathbf{Q} - \mathbf{Q}\overline{u}\right) \tag{2.27}$$

  kde ε je penalizační parametr, jehož volba je velmi důležitá. Obvykle se nastavuje na nějakou malou hodnotu jako např. $\varepsilon = 10^{-9}$. Tato metoda je opět jednoduchá a nemění velikost systému, ale na druhou stranu je závislá na volbě penalizačního parametru a může se stát, že nebude fungovat, pokud systém bude mít velká vlastní čísla. Při výpočtu může dojít, k tomu, že penalizace bude vůči vlastním číslům tak malá, že při výpočtu bude zanedbána. [25]

- **Lagrageovy multiplikátory:** Technika spočívá v rozšíření soustavy o tzv. Lagrageovy multiplikátory λ_i. Každý jeden multiplikátor odpovídá podmínce pro jeden stupeň volnosti. Pro lepší

představu si můžeme uvést příklad, jak obecně vypadá systém rovnic rozšířený o multiplikátory:

$$\begin{pmatrix} \mathbf{K} & \mathbf{Q}^T \\ \mathbf{Q} & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{u} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \overline{\mathbf{u}} \end{pmatrix} \tag{2.28}$$

kde $\mathbf{Ku} = \mathbf{f}$ je původní systém. $\mathbf{Q}$ je binární matice $m \times n$, kde m odpovídá počtu podmínek a m odpovídá počtu vrcholů. Navíc platí, že $\mathbf{Q}_{ij} = 1$ právě tehdy, když i podmínka předepisuje posunutí j vrcholu, jinak $\mathbf{Q}_{ij} = 0$. λ_i je Lagrangův multiplikátor patřící k i podmínce a $\overline{u}$ vektor předepsaných posunutí. Více o Lagrageových multiplikátorech v [26]

Nevýhodou této metody je rozšíření soustavy o počet multiplikátorů a také to, že vlastní čísla, která odpovídají multiplikátorům, můžou být záporná. Na druhou stranu hodnota multiplikátorů bývá označována jako síla působící ve odpovídajících vrcholech při dané deformaci. [25]

2.5 Řešení problému nelineární elasticity

Z předchozích kapitol víme, že pokud chceme řešit deformaci objektu, nejprve sestavíme soustavu rovnic $\mathbf{K}(\mathbf{u}) = \mathbf{f}$ s definovanými okrajovými podmínkami. Tato soustava popisuje systém ve stavu ekvilibria. Obecně lze ale popis tělesa popsat jako:

$$P(\mathbf{u}) = \int_u (\mathbf{K}(\mathbf{u}) - \mathbf{f})\, d\mathbf{u}, \tag{2.29}$$

kde $P(\mathbf{u})$ je potenciální energie deformovaného tělesa, při aktuálním posunutí $\mathbf{u}$.

Z hodin fyziky na střední škole víme, že všechna tělesa v rovnovážném stavu minimalizují svou potenciální energii. Tuto skutečnost musíme promítnout do řešení systému $\mathbf{K}(\mathbf{u}) = \mathbf{f}$. Můžeme to udělat pomocí klasického hledání extrému funkce (globálního minima funkce). Což znamená derivovat funkci potenciální energie podle posunutí a položit ji rovnu 0:

$$\frac{\partial P(\mathbf{u})}{\partial \mathbf{u}} = 0, \tag{2.30}$$

jelikož $P(\mathbf{u})$ je potenciální energie definována rovnicí 2.29, tak minimalizovat potenciální energii tělesa, znamená vlastně řešit

$$\int_u \frac{(K(u) - f)}{\partial u} du = 0 \Rightarrow K(u) - f = 0. \tag{2.31}$$

Vzhledem k nelinearitě systému $K(u) = f$ k řešení využijeme Newton-Raphsonovu metodu, která řeší linearizovaný systém

$$\mathbf{A}(\mathbf{u})\Delta\mathbf{u} = \mathbf{K(u)} - \mathbf{f}, \tag{2.32}$$

kde $\mathbf{A}(\mathbf{u}) = \mathbf{K'}(\mathbf{u})$ je Fréchetova derivace matice pružnosti (tzv. *tangent stiffness matrix*) a $\Delta\mathbf{u}$ je korekce posunutí. V souvislosti s řešením soustavy pomocí Newton-Raphsonovy metody je důležité si uvědomit, že $A(\mathbf{u}) = \mathbf{K'}(\mathbf{u})' = P''(\mathbf{u})$, tzn. že $\mathbf{A}$ lze nahlížet jako Jakobián $\mathbf{K}(\mathbf{u})$ a současně jako Hessián funkce potenciálové energie tělesa $P(\mathbf{u})$. Tuto veličinu je možné spočítat jako analytickou derivaci pružnosti $\mathbf{K}$, anebo jí aproximovat pomocí perturbací (numerická aproximace derivace); v tomto případě pak mluvíme *diskrétní Newton-Raphsonově metodě*.

Ostatní omezení, které musí řešení splňovat jako např. okrajové podmínky zmíněné v kapitole 2.3.6 popř. 2.4.1 nebo i samotné kontakty, které jsou tématem jedné z následujících kapitol práce, jsou aplikovány na linearizovaný systém při řešení pomocí Newton-Raphsonovy metody.

Jak jsme již zmínili v sekci 2.3.4, formulace problému (která přímo určuje, jak se počítají veličiny $\mathbf{K}$ a $\mathbf{A}$ v každém kroku iterace může probíhat dvojím způsobem: v případě varianty *total Lagrangian* je matice $\mathbf{K}$ (a její derivace) spočtena za pomoci 2PK tensoru napětí a všechny veličiny (povrch, objem, normály, tlaky, síly) se počítají v prostorovém nezdeformovaném souřadném systému. Referenční konfigurace je tudíž rovna základní konfiguraci během celého výpočtu. V případě *updated Lagrangian* je matice $\mathbf{K}$ (a její derivace) spočtena pomocí Cauchyho tensoru napětí a veličiny se vyjadřují v konfiguraci, která byla spočtena v předchozí iteraci. Tyto veličiny je tudíž potřeba v každé iteraci přepočítat (odtud *updated* v názvu): referenční konfigurace je dána výsledkem předchozí iterace. Tento přístup je tudíž výpočetně náročnější, nicméně vede na systém rovnic numericky ekvivalentní se systémem, který vznikne v případě použití *total Lagrangian*.

Protože Newton-Raphsonova metoda díky linearizaci garantuje konvergenci pouze v okolí řešení, při aplikaci velkých deformací, kde je základní konfigurace daleko od výsledné, není konvergence zaručena. Z tohoto důvodu se používá různých přídavkových metod (*incremental loading*), čili postupného navyšování působících sil. Tomuto procesu se někdy také říká kvazi-statické řešení. To znamená rozdělení sil na n částí a postupné řešení soustavy vůči předchozímu již vyřešenému stavu. [8, 16, 25]

2.5.1 Přírůstková metoda

Přírůstková metoda patří mezi metody postupného navyšování působících sil. Existuje ve dvou variantách podle velikosti navýšení síly a to ve variantě s konstantním krokem a adaptivním krokem.

Jednodušší varianta metody **přírůstková varianta s konstantním krokem** (*static incremental loading*) aplikuje sílu v n krocích. V každé iteraci i metody, kde $n, i \in \mathbb{N}, 0 < i \leq n$, dochází k aplikaci síly velikosti $\frac{i}{n}F$, kde F je plná velikost působících sil. Pseudokód metody najdete jako algoritmus 1.

Algorithm 1: Přírůstková metoda s konstantním krokem

Data: F reprezentující celkovou působící sílu, n určující počet kroků, ve kterých má být síla aplikována, X_0 počáteční konfigurace tělesa
Result: X konfiguraci tělesa po aplikaci sil

```
1 i = 1 ;
2 X = X_0 ;
3 for i ≤ n do
4 |   F_i = (i/n) F ;
5 |   X = NewtonRaphsonMethod(F_i, X) ;
6 end
7 return X
```

Složitější varianta **přírůstková metoda s adaptivním krokem** (*adaptive incremental loading*), během výpočtu mění počet kroků, po kterých musí dojít k aplikaci celé síly. Změny počtu kroků jsou definovány následujícím způsobem. Pokud Newton-Raphsonova metoda v i ite-

raci metody konverguje v následující $i + 1$ iteraci dochází k snížení počtu kroků na polovinu, pokud nekonverguje metoda se vrací do $i - 1$ kroku a zvýší počet kroků na dvojnásobek. Pseudokód metody najdete jako algoritmus 2.

Algorithm 2: Přírůstková metoda s adaptivním krokem

Data: F reprezentující celkovou působící sílu, n určující počet kroků, ve kterých má být síla aplikována, X_0 počáteční konfigurace tělesa

Result: X konfiguraci tělesa po aplikaci sil

$X = X_0$;
$ratio = 0$;
while $ratio < 1$ **do**
    $ratio = ratio + \frac{1}{n}$;
    $real\_ratio = ratio$;
    if $ratio > 1$ **then**
        $ratio = 1$;
    end
    $F_i = ratio * F$;
    $[X, err] = NewtonRaphsonMethod(F_i, X)$; `// pokud metoda skončí s chybou` X `se nemění a` $err = true$, `jinak` $err = false$
    if err **then**
        $ratio = real\_ratio - \frac{1}{n}$;
        $n = n * 2$;
    else
        $n = \frac{n}{2}$
    end
end
return X

3 Modelování kontaktů

V této kapitole se zaměříme na jeden z cílů práce a to modelování kontaktů deformovatelných těles. V úvodních částech si ukážeme fyzikální vlastnosti těles, které nám pomůžou při matematické definici kontaktů. Pak si popíšeme naivní přístup k modelování kontaktů, který je založen pouze základních definicích vlastností těles (poddajnost) a FEMu. A kapitolu zakončíme definicí tzv. LCP problému, který je základem propracovanějšího algoritmu, jenž se používá na modelování kontaktů v praxi.

3.1 Tuhost vs. poddajnost těles

Před tím než si ukážeme algoritmy, kterými se modelují kontakty těles, měli bychom zmínit veličiny, jež při modelování kontaktů hrají důležitou roli. U deformovatelných těles mluvíme v první řadě o tuhosti (*stiffness*), je schopnost prvku (součásti či konstrukce) přenést zatížení bez porušení. Pro modelování kontaktů je důležitou veličinou tzv. *compliance*, pro kterou budeme v následujícím textu používat český ekvivalent *poddajnost*. Obě veličiny mezi sebou úzce souvisí: matice poddajnosti je homogenní vzhledem k inverzi matice tuhost. Pojďme se nyní podívat na obě vlastnosti trochu podrobněji.

3.1.1 Tuhost

Tuhost tělesa je tudíž fyzikální veličina, která určuje vztah mezi sílou působící na těleso a jeho deformací. Existuje mnoho způsobů, jak definovat tuhost tělesa, ale všechny definice, vycházejí ze vztahu:

$$\text{tuhost} = \frac{\text{činitel}}{\text{deformace}} \tag{3.1}$$

kde činitelem rozumíme např. sílu, tlak nebo kombinace obou působící na těleso. Deformací pak rozumíme rozdíl mezi konfigurací tělesa před působením činitele a konfigurací po ustanovení rovnováhy. K vyjádření tohoto rozdílu můžeme využít deformace, posunutí, úhel nebo změna těchto vlastností.

Z předchozí definice tuhosti lze vidět, že tato není závislá na čase ale pouze na pozici / konfiguraci tělesa a druhu činitelů, které na těleso působí nebo na druhu deformace. Z mnoha druhů tuhosti tělesa zmiňme pouze dvě, a to materiálová tuhost (v inženýrském jazyce též označována jako Youngův modul), která úzce souvisí s Hookovým zákonem zmíněným v 2.3 a jenž lze také definovat jako

$$E = \frac{s}{e} \tag{3.2}$$

kde s je mechanické napětí, e je deformace a E je tuhost materiálu (modul tuhosti).

Pokud do předchozího vztahu navíc dosadíme definice mechanického napětí $s = \frac{F}{S}$ a deformace $e = \frac{\Delta S}{S}$, tak dostáváme jinou definici tuhosti, a to:

$$E = \frac{s}{e} = \frac{\frac{F}{S}}{\frac{\Delta S}{S}} = \frac{F}{\Delta S} \tag{3.3}$$

kde F je síla působící na těleso a ΔS je změna plochy, na níž síla působí. [2]

3.1.2 Poddajnost

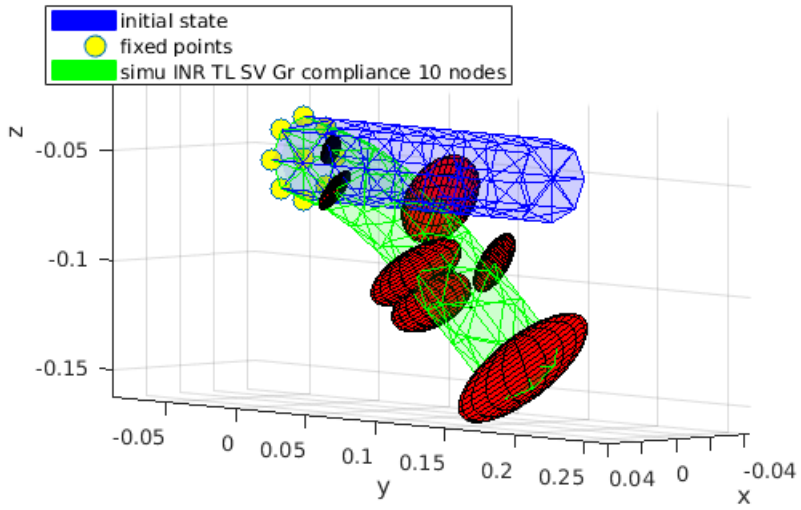
Inverzní vlastností k tuhosti nazýváme poddajností, která vyjadřuje, jak moc je těleso „ochotno“ se hýbat. Matematickou definici poddajnosti je:

$$C = \frac{1}{E} = \frac{e}{s} = \frac{\Delta S}{F} \tag{3.4}$$

kde s je mechanické napětí, e je deformace, F je síla působící na těleso a ΔS je změna plochy, na níž síla působí.

Zamyslíme-li se nad tím, co tato vlastnost znamená, dojdeme k závěru, že obě tyto vlastnosti mají smysl pouze jako vektorové veličiny a má smysl je vyjadřovat stejně jako mechanické napětí či deformaci v tensoru. S využitím znalostí vztahu mezi tensory a elipsoidy (viz. úvodní text ke kapitole 2) si můžeme poddajnost (stejně jako tuhost) vizuálně zobrazit pomocí elipsoidů, které vyjadřují „ochotu“ bodů se hýbat do jednotlivých stran. [7]

Na obrázku 3.1 vidíme ukázku vizualizace poddajnosti válce. Modrý válec znázorňuje počáteční tvar válce, který se působením gravitace zdeformoval do tvaru zeleného válce. Elipsoidy, které vidíme



Obrázek 3.1: Ukázka poddajnosti bodů válce.

uvnitř válce znázorňují velikost poddajnosti vybraných bodů. Je nutné zdůraznit, že bylo vybráno 10 bodů, ale elipsoidy vidíme pouze 7. Důvodem je že 3 elipsoidy, které chybí, patří fixním bodům, jenž jsou zvýrazněny žlutě, a jejich poddajnost je tedy 0. Z obrázku je vidět, že body, které jsou dále od fixních bodů mají větší poddajnost než body blíže, což plně odpovídá definici poddajnosti. Pozorný čtenář namítne, že na obrázku nevidí elipsoidy, ale pouze elipsy, tento fakt je vysvětlen tím, že takto upevněný válec se „ochotněji“ pohybuje vertikálním než horizontálním směrem. Což je určeno geometrií válce, konkrétně poměrem mezi délkou a poloměrem válce. Tento poměr způsobuje, že pro horizontální pohyb (natažení) válce je potřeba mnohem větší síla než k vertikálnímu pohybu (ohýbání).

Z výše uvedených faktů plyne, že poddajnost je určena kombinací dvou faktorů: materiálovými vlastnostmi a geometrií objektu.

3.1.3 Úvod do modelování kontaktů

Modelování kontaktů je v teorii elasticity ztotožněno s definicí tzv. Signoriniho zákona pro dvě tělesa. Ten říká, že dvě tělesa jsou buď v

kontaktu a působí na sebe kontaktními silami, anebo nejsou v kontaktu a kontaktní síly mezi nimi nepůsobí.

Matematicky se definice tohoto zákonu řeší zavedením funkce rozestupu δ (*gap function*) mezi dvěma objekty a kontaktní síly se označují jako λ. S tímto značením pak můžeme Signoriniho problém matematicky popsat jako [1]

$$0 \leq \delta \quad \perp \quad \lambda \geq 0 \tag{3.5}$$

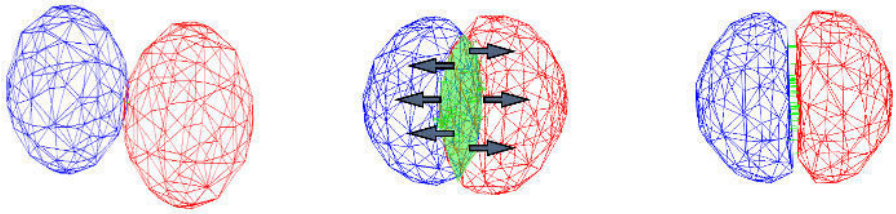
Teorie elasticity obecně uvažuje kontakty mezi n deformovatelnými tělesy, ale v této práci jsou pro jednoduchost uvažovány kontakty mezi deformovatelným a rigidním tělesem. Rigidní těleso je nedeformovatelné těleso, v našem případě rovina. Další zjednodušení, které se v této práci používá, je neuvažování tření mezi tělesy, které ve fyzikálním světě existuje. V teorii elasticity sice existují techniky modelování třecích sil, ale vnášejí do systému další nelinearitu. K těmto technikám patří např. diskretizace pomocí Coulombova kuželu [10].

Jak je vidět z předchozího odstavce, modelování kontaktů je obecně velmi náročný problém, který nemusí mít jednoznačné řešení. V této práci jsou představeny techniky založené na tzv. prediktivním pohybu, kdy algoritmus nejdříve provede krok simulace bez uvažování kontaktů, a poruší tak fyzikální principy a Signoriniho zákon. Následně tuto nefyzikální konfiguraci použije k detekci kolizí, kdy spočítá průnik dvou těles. A výsledek detekce následně použije k spočítání tzv. korektivního pohybu, který zajistí, že výsledná konfigurace splňuje Signoriniho zákon a zároveň nastoluje rovnovážný stav mezi kolidujícími tělesy. Vizuálně zobrazený postup těchto algoritmů vidíme na obrázku 3.2.

3.2 Naivní modelování kontaktů

S využití znalostí definice poddajnosti a znalostí algoritmu pro výpočet deformace tělesa bez kontaktů můžeme odvodit naivní algoritmus výpočtu deformace s kontakty. Naivní algoritmus stejně jako obecně všechny algoritmy, které modelují kontakty, obsahuje tři fáze výpočtu - predikce, detekce a korekce.

Celý výpočet začíná tzv. *prediktivním výpočtem*, při kterém provedeme výpočet aktuálního kroku deformace bez uvažování překážky.



(a) Iniciální konfigurace (b) Prediktivní pohyb a detekce kolizí (c) Korekce

Obrázek 3.2: Tři fáze modelování kontaktů. Převzato z [9].

Což matematicky znamená vyřešit rovnici 2.32:

$$\Delta \mathbf{u} = \mathbf{A}^{-1}[\mathbf{f} - \mathbf{K}] \tag{3.6}$$

$$\mathbf{x}_{free} = \mathbf{x} + \Delta \mathbf{u} \tag{3.7}$$

kde $\Delta \mathbf{u}$ je změna konfigurace tělesa, $\mathbf{f}$ značí aplikované síly, $\mathbf{K}$ je matice tuhosti tělesa, $\mathbf{A}$ je derivace matice tuhosti, $\mathbf{x}$ je předchozí pozice tělesa a $\mathbf{x}_{free}$ je nově vypočítaná pozice tělesa bez uvažování kontaktů.

Následuje *detekce kolize*, při níž hledáme takové body tělesa, u nichž byl porušen fyzikální princip (tzn. došlo např. k průniku tělesa a roviny). V námi uvažovaném případu tělesa a roviny, stačí jednoduše spočítat vzdálenost bodů tělesa od roviny a kontrolovat hodnotu znaménka vzdálenosti pro jednotlivé body. Pokud by totiž došlo k průniku tělesa a roviny, změnilo by se u bodů, které prošly rovinou hodnota znaménka vzdálenosti od roviny. Matematicky vzdálenost bodu o souřadnicích $P = [x_{free}, y_{free}, z_{free}]$ a roviny s rovnicí $\rho : ax + by + cz + d = 0$ můžeme zapsat jako:

$$|P\rho| = \frac{ax_{free} + by_{free} + cz_{free} + d}{\sqrt{a^2 + b^2 + c^2}} \tag{3.8}$$

Poslední částí výpočtu je *korekce*, při ní vhodnou aplikací sil (a tedy vhodným posunutím bodů) napravíme nefyzikální konfiguraci $\mathbf{x}_{free}$. S využitím definice poddajnosti tělesa víme, že vztah mezi silou, poddajností a změnou pozice tělesa je:

$$C = \frac{\Delta S}{F} \rightarrow F = \frac{\Delta S}{C} \tag{3.9}$$

kde C je poddajnost, F je síla a ΔS je změna pozice bodu. Podle tohoto vzorce můžeme pro každý bod spočítat sílu, kterou musíme na bod působit, abychom ho dostali na / nad rovinu. Korekce pak probíhá tak, že ze spočítaných sil vybereme vždy největší sílu a tu aplikujeme v příslušném bodu. Následně iterativně opakujeme tyto tři fáze, dokud všechny body neleží nad nebo na rovině.

Následuje pseudokód celého algoritmu.

Algorithm 3: Naivní algoritmus modelování kontaktů

Input : $\mathbf{X}$ matice s aktuální pozicí bodů, $\mathbf{K}$ je matice tuhosti tělesa, $[a, b, c, d]$ vektor označující normálovou rovnici roviny

Output: $\mathbf{f}_c$ vektor kontaktních sil

```
1  Δu = inv(K') * [f − K]
2  X = X + Δu
3  dist = (ax+by+cz+d) / sqrt(a^2+b^2+c^2)
4  while existuje bod, který je pod rovinou do
5  |  F = dist / inv(K)
6  |  f_c = f_c + max(F) ;   // max(F) vrací vektor s max hodnotou
   |  síly na příslušné pozici
7  |  Δu_corr = inv(K') * [f_c − K]
8  |  X = X + Δu_corr
9  |  dist = (ax+by+cz+d) / sqrt(a^2+b^2+c^2)
10 end
```

Jak z názvu algoritmu plyne, jedná se o jednoduchou variantu řešení, která sebou nese nevýhodu v podobě časové složitosti. Ta je způsobena tím, že při každém průchodu while cyklem se musí znovu sestavit a vyřešit celá MKP.

Navíc pokud budeme uvažovat těleso, které bude dostatečně dlouhé a měkké, může se stát, že algoritmus nikdy neskončí. V krajním případě se totiž u takového tělesa může stát, že korekcí, kterou provádíme ve while cyklu sice vytlačíme některé body nad rovinu, ale body na druhém konci tělesa se působením síly jiné body dostanou pod rovinu. Tento jev se bude opakovat neustále dokola a tak se while cyklus změní na nekonečnou nekonvergující smyčku.

3.3 Algortmus založený na LCP

Jelikož naivní algoritmus modelování kontaktů má mnoho nevýhod a v praxi je nepoužitelný, existuje mnoho jiných algoritmů, které jsou sofistikovanější a méně časově náročné, jelikož nepotřebují v každém kroku znovu aplikovat MKP a řešit celý systém. Jedním z takových algoritmů je algoritmus založený na problému lineární komplementarity (*linear complementarity problem*, LCP) a jeho řešení pomocí Gauss-Seidlovy metody s projekcí.

3.3.1 Problém lineární komplementarity

Problém lineární komplementarity (dále označováno jako LPC) vychází z optimalizačních úloh v matematickém programování. Jedná se o nejjednodušší případ optimalizačního problému za určitých podmínek, který nám dává nástroje jak posléze řešit složitější optimalizační příklady (např. úlohy kvadratického programování, geometrického programování, atd.). Řešení optimalizačních problémů s omezujícími podmínkami má své stěžejní místo v různých oborech od fyziky a matematiky, přes ekonomii až po technické vědy.

Definice 1. ***LCP problém*** *Nechť M je čtvercová matice typu $n \times n$ a $q \in R^n$ je libovolný sloupcový vektor. Najděte vektory $w = (w_1, ..., w_n), z = (z_1, ..., z_n)$, tak aby splňovaly následující vztahy:*

$$w - Mz = q \tag{3.10}$$

$$w, z \geq 0 \tag{3.11}$$

$$w^T z = 0 \tag{3.12}$$

Vstupními parametry problému jsou pak vektor q a matice M, proto můžeme také psát $LCP\,(q, M)$. Triviální řešení tohoto problému je řešení tvaru $w = q, q_i \geq 0 \forall i = 1...n, z = 0$. Ve většině případů nám ale triviální řešení tohoto problému nestačí a tak se pro nalezení netriviálního řešení používají sofistikované metody jako např. Complementary Pivot Method, nebo Gauss-Seidelova metoda projekce, která je popsána v následující kapitole. [23]

3.3.2 Gauss-Seidlova metoda s projekcí

Gauss-Seidelova metoda s projekcí (*projected Gauss-Seidel method*) je pokročilým nástrojem pro řešení soustavy lineární rovnic $Ax = b$, kde A je regulární čtvercoví matice řádu n, x je vektor neznámých a b je vektor pravé strany. Jedná se o iterační metodu, která nejprve převede soustavu do iteračního tvaru:

$$x = Cx + d \tag{3.13}$$

kde x je vektor neznámých, C je iterační matice a d je sloupcový vektor. Tuto transformaci můžeme jednoduše provést tak, že si z každé rovnice soustavy vyjádříme jednu neznámou.
Následně rovnici 3.13 přepíšeme na iterační předpis s tím, že při výpočtu i neznámé $1 < i \leq n$, kde n je počet neznámých, využijeme znalostí předchozích vypočtených neznámých [22]. Matematicky můžeme tento vztah zapsat jako:

$$x_i^{(k+1)} = \frac{1}{c_{ii}} \left(d_i - \sum_{j=1}^{i-1} c_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} c_{ij} x_j^{(k)} \right), i = 1, ...n \tag{3.14}$$

3.3.3 Popis algoritmu

Nyní, když už víme, co je to LCP problém a víme jak vypadá Gauss-Seidelova metoda, využijeme tyto znalosti a ukážeme si jak vypadá algoritmus pro modelování kontaktů s využitím LCP problému. Výpočet algoritmu můžeme opět rozdělit do tří fází – predikce, detekce a korekce.

První fáze *predikce* je naprosto stejná jako u naivního algoritmu tzn. dochází k spočítání aktuální pozice tělesa bez uvažování překážky, což matematicky znamená vyřešení systému s pomocí Newton-Raphsonovy metody a získání pozice tělesa $\mathbf{x}_{free} = [x_{free}, y_{free}, z_{free}]$:

$$\Delta \mathbf{u} = inv(\mathbf{A}) * [\mathbf{f} - \mathbf{K}] \tag{3.15}$$

$$\mathbf{x}_{free} = \mathbf{X} + \Delta \mathbf{u} \tag{3.16}$$

V druhé fázi *detekce* hledáme body, které prošly rovinou, tzn. body jejichž vzdálenost od roviny změnila znaménko. Předpokládejme nyní, že všechny body tělesa v klidovém stavu měly vzdálenost od

roviny kladnou. Potom lze detekci pro každý jednotlivý bod $P = [x_{free}, y_{free}, z_{free}]$ matematicky napsat jako řešení následující nerovnice:

$$ax_{free} + by_{free} + cz_{free} + d < 0 \tag{3.17}$$

kde rovinu se kterou tělesa koliduje vyjadřujeme jako $\rho : ax + by + cz + d = 0$.

Nyní si označme n jako počet bodů kolidujících s rovinou podle předešlé nerovnice a $\mathbf{Q} = X_1 ... X_n$ jako množinu těchto bodů. S využitím rovnice 3.8 pro každý z těchto bodů spočítáme jeho vzdálenost od roviny d_i, kde $i \in N, 0 < i \leq n$. Při určování znaménka vzdálenosti využijeme následující konvenci:

- $d_i < 0$ pokud bod X_i prošel rovinou (nerespektuje překážku)
- $d_i \geq 0$ pokud je bod X_i nad nebo na rovině

V poslední fázi *korekce* využijeme definici LCP problému a jeho vyřešení pomocí Gauss-Seidelovy metody. Nejprve musíme v problému, který chceme řešit, najít LCP problém. To ale není vůbec složité, stačí si napsat k sobě rovnice, které plynou ze Signoriniho zákona, definovaného v 3.1.3. A vypadne nám následující soustava rovnic, která je velmi podobná soustavě rovnic LCP problému.

$$dQ = dQ + W(Q, P) \cdot fP \tag{3.18}$$
$$fQ \geq 0 \tag{3.19}$$
$$dQ \geq 0 \tag{3.20}$$
$$fQ^T dQ \geq 0 \tag{3.21}$$

kde Q je množina bodů, které prošly rovinou, P je množina bodů na / nad rovinou, dQ, dP pak značí vzdálenost bodu od roviny (kladná pokud je bod nad rovinou a záporná pod rovinou), fQ, fP jsou síly působící na body a $W(Q, P)$ je **Delassův operátor**.

Delassův operátor W („kondenzovaná“ poddajnost) je matice velikosti $n \times n$, kde w_{ij} udává vztah mezi body s indexy i, j, určený tím, že se oba body nacházejí v jednom tělese. K jeho výpočtu je potřeba mít matici tuhosti C a normály jednotlivých bodů n_i. Hodnota Delassova operátoru pro body s indexy i, j se spočítá:

$$w_{ij} = n_i' \cdot C(i, j) \cdot n_j \tag{3.22}$$

kde $C(i,j)$ je 3×3 matice s indexy odpovídajícími bodům X_i a X_j.

K řešení této soustavy rovnic použijeme tzv. Gauss-Seidlovu metodu s projekcí (algoritmus 4). Ta pro každé omezení (bod Q, který je pod rovinou) hledá sílu f_q, aby platil Signoriniho zákon. Navíc musí platit, že vztah mezi silou působící na bod a vzdáleností, o kterou je potřeba bod posunout je přímo úměrný poddajnosti.

Algorithm 4: Gauss-Seidlova metoda s projekcí

Data: W je Delassův operátor, d vektor iniciálních vzdálenosti bodů od roviny se znaménkovou konvencí, Q množina bodů pod rovinou

Result: f vektor působících sil v bodech

```
1  nastavení TOL jako tolerance metody
2  while není dosažen max. počet iterací do
3  |  error = 0 ;
4  |  for q ∈ Q do
5  |  |  errQ = f_q ;
6  |  |  for p ∈ Q − q do
7  |  |  |  d_q = d_q + W(q, p) ∗ f_p;
8  |  |  end
9  |  |  f_q = f_q − d_q/W(q, q) ;
10 |  |  if f_q < 0 then
11 |  |  |  f_q = 0;
12 |  |  end
13 |  |  error = error + abs(W(q, q) ∗ (f_q − errQ);
14 |  end
15 |  if error < TOL then
16 |  |  return f;
17 |  end
18 end
```

Poslední a nejdůležitější částí korekce je aplikace spočtených sil f na systém. Což se provedeme tak, že síly f promítneme do globálního silového vektoru (normálový vektor sil se všemi body systému). Což je jen složitý název pro složení vektoru, který obsahuje pro jednotlivé body normálu vynásobenou korekční silou. Pokud k danému bodu korekční síla není, je normála vynásobena nulou. Označme takový

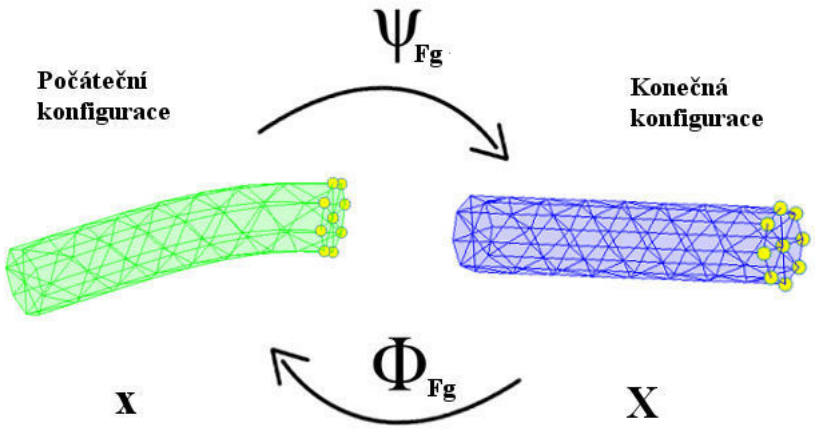
vektor λ. Následně je spočítána korekce pro celý systém a to vyřešením rovnice:

$$\Delta \mathbf{u}_{corr} = \mathbf{A}^{-1}\lambda \tag{3.23}$$

a korekce připočítána k aktuální poloze systému $\mathbf{x} = \mathbf{x} + \Delta \mathbf{u}_{corr}$. [9, 10]

4 Algoritmus kompenzace sil

Problém dnešních modelů orgánů používaných při různých operačních zákrocích přichází již při samotné tvorbě modelu. Ten je vystaven na datech o pacientovi, které získají lékaři při skenech pacienta. Tyto informace jsou o orgánech, které nejsou v klidové pozici, protože na ně působí síly např. gravitační síla nebo tlaková síla způsobená umístěním orgánů v uzavřené tekutině (např. tlaky působící na mozek, způsobené umístěním mozku v mozkomíšní tekutině, která je uzavřena v lebce). Ačkoliv jsou tedy anatomie rekonstruované z obrazů nasnímaných např. pomocí CT nebo MRI, již zdeformované a tak jejich iniciální mechanické napětí je nenulové, většina simulačních algoritmů tento fakt zanedbá a uvažuje tento stav za klidový. Algoritmus kompenzace sil představený v této kapitole otevírá možnost rekonstrukce klidového stavu tělesa při znalosti deformované pozice tělesa a hodnot působících sil. [28]



Obrázek 4.1: Klasická deformace vs. kompenzace sil

Na obrázku 4.1 vidíme porovnání klasického problému deformace tělesa (směr funkce ϕ) a problému kompenzace sil (směr funkce ψ).

4.1 Naivní přístup k řešení problému

V rámci naivního přístupu k problému lze jednoduše vytvořit iterativní algoritmus řešení problému kompenzace sil. Algoritmus začíná tím, že počáteční, deformovanou konfiguraci tělesa X^{init} nastavím jako počáteční odhad konečné, nedeformované konfigurace tělesa X^0. Provedu přímou simulaci a uložím si rozdíl mezi výsledkem přímé simulace a počáteční konfigurací tělesa u^0. Pokud nebyl dosažen maximální počet iterací, nebo pokud jsem nedosáhl dostatečné přesnosti mezi výsledkem přímé simulace a známou konfigurací tělesa X^{init}, tak je upraven odhad nedeformované pozice tělesa $X^j = X^{j-1} - u^{j-1}$. Potom se opět provede přímá simulace a znovu spočítá rozdíl u^j, chyba a navýší se hodnota počtu iterací. Lepší představu o algoritmu si můžete udělat s pomocí následujícího pseudokódu. [11, 12, 28]

Algorithm 5: Iterativní algoritmus pro výpočet kompenzace sil

Data: X^{init} je deformovaná konfigurace tělesa
Result: X^{j-1} je odhadnutá nezdeformovaná konfigurace tělesa

$X^0 = X^{init}$;
spustíme přímou simulaci 0 s počáteční konfigurací X^0 ;
$u^0 = x^0 - X^0$;
$err =$ chyba mezi x^0 a X^0;
$j = 1$;
while $err > \varepsilon$ & $j < NB^{max}$ **do**
    $X^j = X^{j-1} - u^{j-1}$;
    spustíme přímou simulaci 0 s počáteční konfigurací X^{j-1} ;
    $u^j = x^j - X^j$;
    $err =$ chyba mezi x^j a X^{init};
    $j = j + 1$;
end
return X^{j-1};

4.2 Algoritmus využívající „inverzi“

V kapitole 2.3 byla popsána metoda výpočtu deformované pozice tělesa se znalostí sil působících na dané těleso. Při takto definovaném výpočtu jsme si zavedli následující značení, X pro nezdeformovanou pozici tělesa a x pro zdeformovanou pozice tělesa. Tohoto značení se budeme držet i nadále.

Zamyslíme-li se nad rozdílem mezi problémem definovaným v 2.3.2 a problémem, který řešíme v této kapitole, dojdeme k závěru, že oba problémy se liší pouze prohozením znalostí / neznalostí x a X. Tento fakt se nejvíce odrazí na rovnici 2.16, kde je deformační napětí je definováno jako funkce proměnné X, kterou ale zde neznáme. Tento malý rozdíl vede k nutnosti udělat několik modifikací v předchozím algoritmu, abychom ho mohli použít při řešení inverzního problému.

První změna, kterou musíme provést vychází z odlišné definice posunutí u jako

$$u'(x) = X - x \tag{4.1}$$

kde X je nezdeformovaná a x je zdeformovaná pozice tělesa, u' označíme posunutí příslušící inverznímu problému. S touto novou definicí posunutí je spojena i nová definice deformačního gradientu F_{inv}. Z obrázku 4.1 na začátku této kapitoly a ze znalosti definic deformačního gradientu a deformační funkce z kapitoly 2.3.2 můžeme definovat vztah mezi deformačním gradientem F klasického deformačního výpočtu a F_{inv}.

$$F_{inv} = \frac{\partial \psi(x)}{\partial x} \tag{4.2}$$

$$F = \frac{\partial \phi(X)}{\partial X} = \frac{\partial \phi(\psi(x))}{\partial \psi(x)} = \frac{\partial x}{\partial \psi(x)} = F_{inv}^{-1} \tag{4.3}$$

kde ψ, ϕ jsou deformační funkce, x, X jsou zdeformovaná a nezdeformovaná pozice tělesa. Pro lepší představu vše označeno podle obrázku na začátku této kapitoly. Z obrázku taky vidíme, že $\psi = \phi^{-1}$. Z rovnice 4.3 plyne, že při řešení inverzního problému můžeme deformační gradient počítat stejným způsobem jako u klasické deformace, jen pak pouze místo F použijeme F^{-1}.

Dalším rozdílem mezi výpočtem klasického a inverzního deformačního problému je v metodě samotného výpočtu. Klasický problém

můžeme vyřešit pomocí dvou definovaných metod v kapitole 2.4: total a updated Lagrangians. Při uvažování inverzního problému se ukazuje, že nalezení nedeformované konfigurace může být formulováno jako kombinace těchto dvou metod. Základní metodou výpočtu je total Lagrangian. Ale protože známe síly působící na konečnou konfiguraci tělesa, jejíž geometrii neznáme, podle kapitoly 2.1 se jedná o Eulerovský pohled, který využívá Cauchyho napětí, tak je při řešení inverzního problému v rovnici 2.18 nahrazeno 2PK napětí Cauchyho mechanickým napětím. Externí síly působící na těleso jsou nahrazeny silami v deformovaném stavu.

Konstituční rovnice, kterou potřebujeme vyřešit, pak vypadá takto:

$$0 = \frac{1}{|F_{inv}|} f(x) + \nabla \cdot \sigma(x), x \in \Omega \tag{4.4}$$

kde F_{inv} je deformační gradient definovaný v 4.2, f jsou síly působící na těleso, σ vyjadřuje Cauchyho napětí.

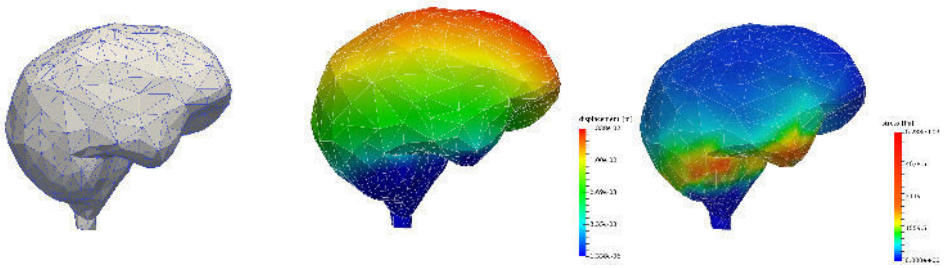
4.3 Ilustrace výsledků algoritmu na reálných datech

V úvodu této kapitoly byl uveden příklad s modelováním mozku, na který působí tlaky, protože je obklopen mozkomíšním mokem (CSF) a uzavřen v lebce. Algoritmus představený v této kapitole můžeme použít, pokud chceme zjistit, jak vypadá rozložení pole mechanického napětí způsobeného tlakem CSF na uvedený orgán. Tato informace je důležitá pro odhad posunutí mozku (tzv. *brain-shift*), ke kterému dochází po vytvoření otvoru do lebky při operacích mozku.[4]

Simulace byla provedena s následujícím nastavením parametrů: pro modelování je použit Mooney-Rivlinův hyperelastický materiál s logaritmickou funkcí penalizace změny objemu s parametry $C_{01} = 3922\,\text{Pa}$ a $C_{10} = 30.838\,\text{Pa}$ [31]. Tlak CFS na mozek se obvykle udává jako $7 - 15\,\text{mmHg}$, což je přibližně $900 - 2000\,\text{Pa}$. Simulace pracuje s dolní hranicí tohoto intervalu, a tak je hodnota nastavena na $1000\,\text{Pa}$. Aplikace okrajových podmínek byla provedena „uchycením“ spodní části mozku pomocí homogenní Dirichletovy okrajové podmínky. Toto zjednodušení anatomicky odpovídá spojení mozku a prodloužení míchy.

Provedení simulace pak zahrnovalo dva následující kroky:

- **Kompenzace tlaku** provedená s využitím zpětné simulace (undated_inverse), jejíž výsledkem je konfigurace bez aplikovaného tlaku. Z obrázku 4.2b vidíme, že maximální posunutí mozku je do 1.4 cm. Tato konfigurace je, ale pouze „teoretická“, protože i kdybychom vypustili veškerou CSF, tak by mozku k takové „deformaci“ bránila lebka. Tu ale neuvažujeme, protože by znamenala aplikaci kontaktů, které nejsou v metodě zatím aplikovány.
- **Aplikace přímé simulace**, kterou spustíme z konfigurace spočteme v předchozím kroku. Výsledkem je síť, která je geometricky stejná s původní sítí mozku, ale obsahuje informaci o rozložení tlaku (obrázek 4.2c), kterou můžeme využít k dalším výpočtům.



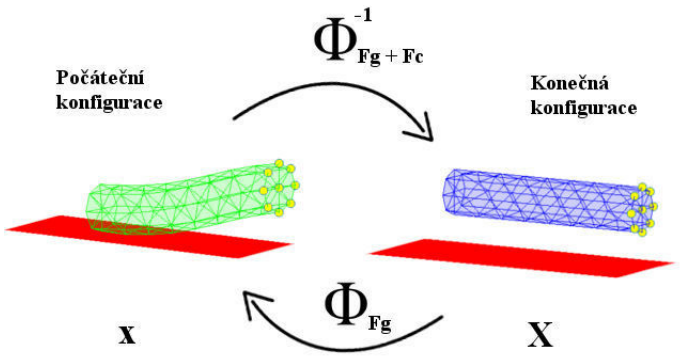
(a) počáteční geometrie (b) posunutí geometrie bez CFS tlaku (c) geometrie s distribucí napětí

Obrázek 4.2: Ukázka simulace na datech mozku.

Simulace byla spouštěna na počítači PC Intel Core i7-6700K @ 4.00 GHz, 16 GB RAM s Matlabem verze R2016a (9.0). Síť použitá při simulaci byla složena z 1916 elementů (1830 stupňů volnosti). Zpětný krok simulace (provedený metodou updated Lagrangian s inverzí) trval 214 s a přímý krok (provedený pomocí metody total Lagrangian) 94.7 s.

5 Kontakty v kompenzaci sil

V předchozí kapitole byl představen algoritmus pro rekonstrukce klidového stavu tělesa s využitím znalostí sil působících na těleso. Tento algoritmus je velmi perspektivní pro využití při simulacích v chirurgii. Při uvažování použití algoritmu při modelování reálných simulací v medicíně, narážíme na problém se znalostí všech působících sil. Při modelování orgánů musíme mezi působící síly, uvažovat i kontaktní sily, kterými na sebe navzájem orgány lidského těla působí a jejichž velikost neznáme. Z kapitoly 3 známe metody modelování kontaktů při přímém algoritmu, cílem této kapitoly je blíže prozkoumat chování kontaktů při kompenzaci sil a případně navrhnout algoritmus odhadu kontaktních sil při kompenzaci sil.



Obrázek 5.1: Ilustrace hledaní nezdeformované konfigurace.

Na obrázku 5.1 vidíme problém znázorněný graficky. Obrázek vlevo (počáteční konfigurace) znázorňuje těleso deformované gravitací a kontaktními silami, obrázek vpravo představuje těleso v klidovém stavu – bez působení gravitace a kontaktních sil (konečná konfigurace). Funkce Φ^{-1}_{Fg+Fc} je inverzní deformační funkce, která závisí na gravitaci a kontaktních silách, Φ_{Fg} označuje deformační funkci díky gravitaci.

V této kapitole se budeme držet značení definovaného v kapitole 2.3.

5.1 Definice problému

Problém představený v úvodu této kapitoly přesahuje rámec diplomové práce a tak se budeme zabývat pouze kontakty jednoho deformovatelného tělesa s nehybnou překážkou - rovinou. Na vstupu máme těleso, které bylo zdeformováno vnějšími silami f, ale protože během deformace těleso interagovalo s překážkou, působí na něj během deformace nejen původní vnější síly ale také kontaktní síly f_c. Našim úkolem je ze znalosti zdeformované pozice tělesa x, působících sil f a umístění překážky určit nedeformovanou pozici tělesa X. Síly působící na těleso ve stejném okamžiku, můžeme jednoduše sečíst, označme tedy všechny síly působící na těleso $f_{all} = f + f_c$. Protože se jedná o problém kompenzace sil, definovaný v kapitole 4 využijeme rovnici 4.2 a přepíšeme ji do tvaru s kontakty.

$$0 = \frac{1}{|F_{inv}|} f_{all}(x) + \nabla \cdot \sigma(x), x \in \Omega \tag{5.1}$$

kde F_{inv} je deformační gradient definovaný v 4.2, f_{all} jsou všechny síly působící na těleso, σ vyjadřuje Cauchy tensor mechanického napětí.

Protože se jedná o stejný problém jako problém v předchozí kapitole, můžeme k řešení využít algoritmus kompenzace sil popsaný v kapitole 4.2. Jedinou překážkou k jeho využití je neznalost kontaktních sil a právě jejich odhadem se bude zabývat v následujícím textu.

5.2 Odhad kontaktních sil

Kontaktní síly působící na těleso během interakce s pevnou překážkou neumíme při znalosti pouze deformované pozice tělesa spočítat přímo a tak se pokusíme nalézt iterativní algoritmus odhadu kontaktních sil.

5.2.1 Iterativní algoritmus s postupným vylepšováním

Iterativní algoritmus odhadu kontaktních sil je založen na postupném vylepšování odhadu sil, které vede k minimalizaci chyby, která vznikne aplikací těchto sil. Obecný návrh takového algoritmu představuje algoritmus 6.

Z pseudokódu obecného iterativního algoritmu vidíme, že výpočet závisí na vhodné volbě funkcí opravy kontaktních sil na řádku 5 a

Algorithm 6: Obecný iterativní algoritmus odhadu F_c

Data: X^{def} je deformovaná konfigurace tělesa, F jsou vnější síly působící na těleso
Result: F_c jsou kontaktní síly působící na těleso

```
1  repeat
2      if j == 0 then
3          F_c ← nastavení počátečního odhadu sil ;
4      else
5          F_c ← Oprava_kontaktnich_sil(F_c, err) ;
6      end
7      err = Vypocet_chyby(F_c, F, X^def);
8      j = j + 1;
9  until err > ε & j < NB^max;
10 return F_c;
```

výpočtu chyby na řádku 7. Vhodným navržením těchto funkcí se budeme zabývat následující textu práce.

Výpočet chyby

Při výpočtu chyby byly uvažovány dva přístupy – postupné vypínání sil a vypínání všech sil.

První varianta algoritmu, která využívala postupného vypínání sil, má za cíl spočítat velikost kontaktních sil bez nutnosti používat algoritmy uvažující kontakty, které jsou obecně pomalejší než ostatní používané algoritmy. Tento přístup k řešení problému naráží na problém s nelinearitou samotného problému, která znemožňuje rozložit rovnici 5.1 na postupné řešení rovnic

$$0 = \frac{1}{|F_{inv}|} f(x) + \nabla \cdot \sigma(x), x \in \Omega \tag{5.2}$$

$$0 = \frac{1}{|F_{inv}|} f_c(x) + \nabla \cdot \sigma(x), x \in \Omega \tag{5.3}$$

kde F_{inv} je deformační gradient definovaný v 4.2, f jsou síly působící na těleso (např. gravitace), f_c jsou kontaktní síly, σ vyjadřuje Cauchy tensor mechanického napětí.

Algorithm 7: Výpočet chyby s postupným vypínáním sil

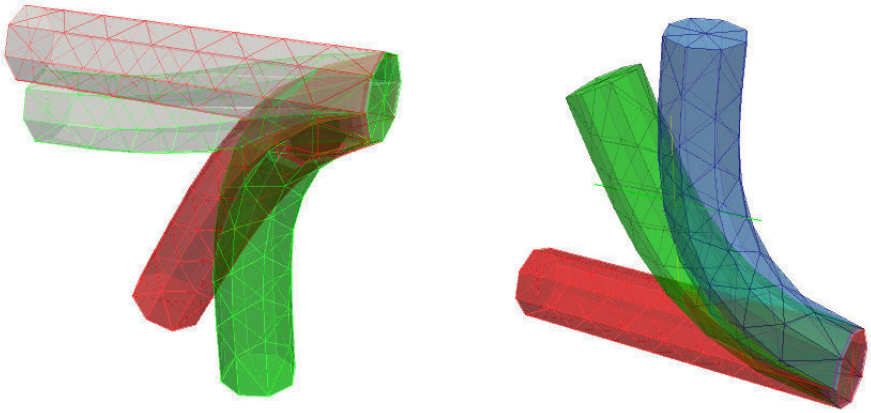
Data: X^{def} je deformovaná konfigurace tělesa, F jsou vnější síly působící na těleso, F_c odhad kontaktních sil
Result: *err* chyba při aplikaci daných F_c

1 $X^{init} \leftarrow$ spustíme inverzní simulaci s počáteční konfigurací X^{def} a vypneme F ;
2 $X \leftarrow$ spustím inverzní simulaci z X^{init} a vypnu F_c ;
3 $Y \leftarrow$ spustím inverzní simulaci z X^{def} a vypnu F_c ;
4 $x \leftarrow$ spustím klasickou simulaci z X s F ;
5 $err = \sum_{i \in nodes} \sqrt{\left(x_1^i - Y_1^i\right)^2 + \left(x_2^i - Y_2^i\right)^2 + \left(x_3^i - Y_3^i\right)^2}$;
6 **return** *err*;

Algoritmus 7 představuje možnou variantu výpočtu chyby s postupným vypínáním sil. Tento algoritmus nejprve algoritmem kompenzace sil spočítá pozici tělesa bez působení sil mimo kontaktních sil (řádek 1), vypne odhadované kontaktní síly (řádek 2) a následně spustí klasickou deformaci tělesa bez uvažování kontaktů (řádek 4). Výslednou pozici tělesa pak porovnává s pozicí, kterou dostaneme, když algoritmem kompenzace sil z deformované pozice tělesa spočítáme pozici tělesa bez kontaktních sil (řádek 3 a 5).

Jak bylo zmíněno v úvodu této kapitoly, hlavním důvodem k sestavení tohoto algoritmu byla snaha vyhnout se počítání dopředného algoritmu s kontakty, které je v porovnání s algoritmem kompenzace sil či dopředným algoritmem bez kontaktů pomalejší. Toto řešení výpočtu chyby není možné, protože naráží na problém s nelinearitou. Jako příklad si uveďme obrázek 5.2 s výsledky simulací při postupném vypínání sil.

Obrázek 5.2 zobrazuje 2 situace, které dokumentují problém s nelinearitou. Na obrázku 5.2a je znázorněno porovnání výsledků dvou simulací. První simulace, jejíž iniciální stav je zeleno-šedý a konečný stav je vykreslen zelenou barvou, znázorňuje situaci, kdy u tělesa zdeformovaného pomocí gravitace a kontaktů spustíme inverzní algoritmus a vypneme kontaktní síly. Druhá simulace znázorněna červenou barvou (iniciální stav červeno-šedě a konečný červeně) ukazuje aplikaci gravitace na nezdeformovanou pozici tělesa bez kontaktních



(a) Vypnutí kontaktních sil (b) Postupné vypínání sil

Obrázek 5.2: Ukázky nelinearity problému

sil. Kdyby daný problém nebyl nelineární červená a zelené pozice tělesa by byly totožné. Z obrázku ale vidíme, že tomu tak není.

Na obrázku 5.2b je zachyceno postupné vypínání gravitace a kontaktních sil. Červené těleso znázorňuje těleso, na které nepůsobí žádné síly, a tedy také pozici těleso, do níž se potřebujeme postupným vypínáním sil dostat. Modré těleso ukazuje pozici, do níž se těleso dostane pokud v zdeformovaném stavu vypneme gravitační sílu a zelené těleso je pozice tělesa u kterého jsme vypnuli nejprve gravitaci a poté kontaktní síly. Protože červené těleso není totožné se zeleným, nemůžeme tento rozklad sil využívat.

Jelikož se ukázalo, že algoritmus postupného vypínání sil není možné aplikovat, byla vytvořena jiná metoda výpočtu chyby, která obsahuje dopředný algoritmus s počítáním kontaktů – algoritmus chyby bez postupného vypínání sil.

Algoritmus 8 představuje výpočet chyby bez postupného vypínání sil a je založen na počítání klidové pozice tělesa bez působení sil, vypnutím všech sil působících na těleso tzn. i odhadu kontaktních sil (řádek 1). Na řádku číslo 2 je z této klidové pozice spuštěn klasické výpočet s uvažování překážky, abychom získali zdeformovanou pozici tělesa při působení odhadnutých kontaktních sil. Jako chyba odhadu je označen součet euklidosvkých vzdáleností všech odpovídajících

Algorithm 8: Výpočet chyby bez postupného vypínání sil

Data: Y je deformovaná konfigurace tělesa, F jsou vnější síly působící na těleso, F_c odhad kontaktních sil

Result: *err* chyba při aplikaci daných F_c

1 $X \leftarrow$ spustíme inverzní simulaci s počáteční konfigurací Y a vypneme $F - F_c$;

2 $x \leftarrow$ spustím klasickou simulaci s překážkou z X ;

3 $err = \sum_{i \in nodes} \sqrt{\left(x_1^i - Y_1^i\right)^2 + \left(x_2^i - Y_2^i\right)^2 + \left(x_3^i - Y_3^i\right)^2}$;

4 **return** *err*;

si bodů původní deformované konfigurace tělesa Y a deformované konfigurace při odhadnuté klidové pozici x.

Oprava kontaktních sil

Funkce opravy kontaktních sil je založena korekční části modelování kontaktů představených v kapitole 3.3.3. Využívá Gauss-Seidelovu metodu s projekcí a jako vstupní hodnota vzdálenosti je nastavena záporná hodnota chyby.

Při testování metody bylo zjištěno, že metoda konverguje velmi pomalu a je tak nepoužitelná. Což možná souvisí s tím, že opravování sil je prováděno na základě vzdáleností v deformované pozici tělesa. Možná by stálo za úvahu převést velikost těchto sil pomocí deformačního gradientu do nezdeformovaných souřadnic.

5.2.2 Algoritmus založený na minimalizaci chyby

Tento iterativní algoritmus je založen na odhadu sil za pomocí minimalizace chybové funkce. Kód funkce je popsán pseudokódem algoritmu 6 s definicí chyby podle algoritmu 8, ale opravu sil necháváme počítat Matlabem voláním funkce *fminsearch*. Ta je funkcí z optimalizačního balíku Matlabu a je založena na hledání minima funkce díky numerické aproximaci derivace funkce.

Jelikož algoritmus využívá chybu definovanou podle algoritmu 8, musí algoritmus pracovat s funkcemi *runInverseSimulation.m* a *run-*

DirectSimulation.m, které spouštějí inverzní nebo přímou simulaci s danými parametry s vhodně nastavenými parametry pro výpočet.

```
1 function  runInverseSimulation(params, model, fem, dirichlet, solver, configuration,
      outDir, k, l)
2   params.outDir = sprintf('%s/inverse_%d_%d', outDir,k,l);
3   params.workingDir = params.outDir;
4   solver.numIncSteps=4;
5   solver.newtonMaxIt=10;
6   createWorkingDirectory(params);
7   staticIncrementalLoading(params, model, fem, dirichlet, solver, configuration);
8 end
```

Obrázek 5.3: Kód funkce runInverseSimulation.m

Obrázek 5.3 ukazuje kód funkce, která spouští inverzní algoritmus s odhadem kontaktních sil. Funkce obsahuje spuštění inverzní simulace na řádku číslo 7, přenastavení parametrů řešiče (řádky 4 – 5) a vytvoření složky pro uložení výsledku simulace (řádky 2, 3, 6).

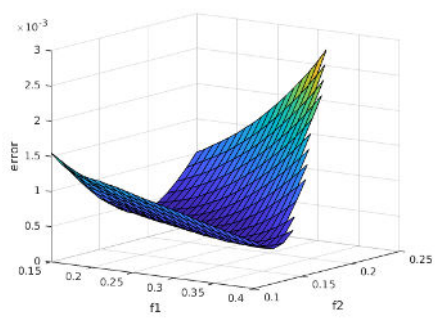
```
1  function [finalNodalPosition, conf] = runDirectSimulation(params, outDir,k,l)
2    params.vtkVolumeFile = sprintf('%s/inverse_%d_%d/volFinal.vtk', params.experimentName,
         k, l);
3    params.vtkSurfaceFile = sprintf('%s/inverse_%d_%d/surfFinal.vtk', params.
       experimentName, k, l);
4    params.lagrangian=Lagrangian.total;
5    params.solutionMethod=SolutionMethod.INR;
6    params = checkConfig(params);
7    params.outDir = sprintf('%s/direct_%d_%d', outDir,k,l);
8    params.workingDir = params.outDir;
9    [model, dirichlet] = setupModel(params);
10   fem = setupFEM(model);
11   solver = setupSolver(params, model);
12   configuration = setupConfiguration(model, dirichlet);
13   createWorkingDirectory(params);
14   initDir=cd(params.workingDir);
15   if (params.doProfile)
16     profile on;
17   end
18   solver.numIncSteps=10;
19   solver.newtonMaxIt=1;
20   conf = loadingWithPlaneContact(params, model, fem, dirichlet, solver, configuration);
21   [finalNodalPosition,~,~] = readVTK(sprintf('%s/volFinal.vtk', params.outDir));
22 end
```

Obrázek 5.4: Kód funkce runDirectSimulation.m

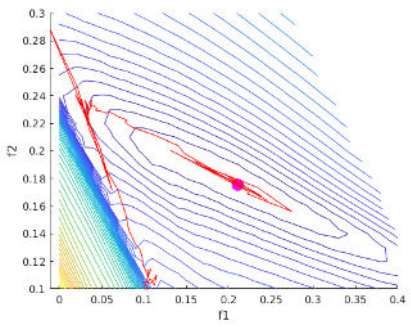
Obrázek 5.4 předkládá kód funkce *runDirectSimulation.m*. Z kódu vidíme, že dochází k přenastavení parametrů, jako vhodný výběr

souborů s geometrií tělesa (řádky 2 a 3), nastavení vhodné metody řešení (řádky 4 – 5), nastavení adresáře pro uložení výsledků (řádky 7 a 8) a spuštění výpočtu (9 – 20). Návratová hodnota funkce je matice pozice pro jednotlivé body a struktura s konfiguračním nastavením.

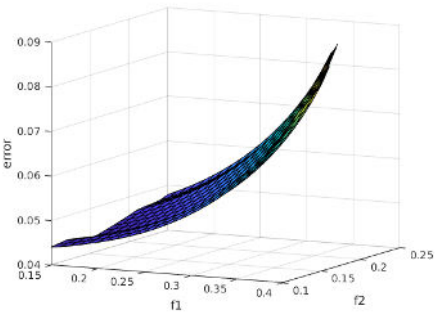
Na řádcích 18 a 19 je provedeno nastavení maximálního počtu iterací Newton-Raphsonovy metody z důvodu urychlení výpočtu a lepších vlastností funkce odhadu chyby. Rozdílné chování funkce chyby pro výpočty s použitím klasické a jedno iterační Newton-Raphsonovy metody při výpočtu simulací vidíme na obrázku 5.5 .



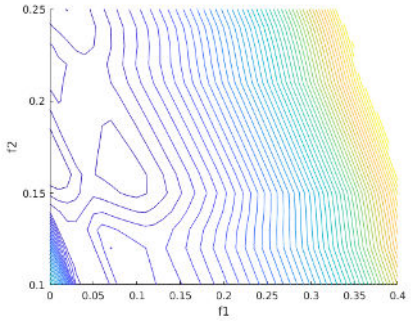
(a) počet NR iterací = 1



(b) počet NR iterací = 1



(c) počet NR iterací = 15



(d) počet NR iterací = 15

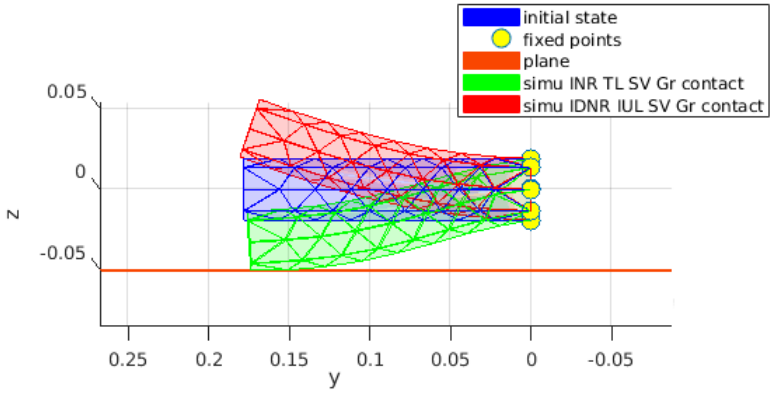
Obrázek 5.5: Grafy funkce chyby pro různé metody řešení v závislosti na maximálním počtu iterací Newton-Raphsonovy metody.

Obrázky 5.5 vykreslují graf (klasický 3D a obrysový graf) chyby pro deformace tělesa (cylinder složen ze 152 elementů), které deformujeme gravitací a při kontaktu s rovinou se roviny dotýká ve dvou bodech.

Obrázky 5.5a a 5.5b zobrazují hodnoty funkce počítající chybu s využitím pouze 1 iterace Newton-Raphsonovy metody pro hodnoty okolí skutečných hodnot sil spočítaných simulaci kontaktu tělesa s překážkou ($f_1 = 0.335756, f_2 = 0.128877$). Z obrázků vidíme, že tvar funkce je vhodný pro aplikování optimalizačních funkcí. Obrázek 5.5b obsahuje znázornění výpočtu minimalizace chyby provedené pomocí minimalizační funkce s počáteční hodnotou sil $[0.1, 0.1]$.

Obrázky 5.5c a 5.5d ukazují hodnoty funkce počítající chybu s využitím konvergující Newton-Raphonovy metody. Obrázek nám ukazuje jiné chování funkce než v předchozím případě a navíc vidíme, že funkce nedosahuje minimální hodnoty v očekávaných hodnotách sil.

Na závěr kapitoly ještě uveďme výsledky simulací při použití algoritmu s minimalizací chyby. Na příkladu válce deformovaného gravitací.



Obrázek 5.6: Výsledek algoritmu minimalizace chyby

Na obrázku 5.6 vidíme porovnání výsledné pozice válce spočítané daným algoritmem (červený válec) se skutečnou klidovou pozicí (modrý válec). Toto porovnání můžeme provést, protože máme implementovány dopředný i zpětný algoritmus s kontakty. Z obrázku je vidět, že algoritmus nenašel pravou klidovou pozici. Nicméně

porovnáme-li deformovanou pozici tělesa, kterou jsme dostali na vstupu s deformovanou pozicí, kterou dostaneme při spuštění přímé simulace s odhadnuté klidové pozice dostáváme dva téměř totožné výsledky.

V této kapitole byl představen problém inverzního algoritmu při uvažování kontaktů. Ukázali jsme, že tento problém se při znalosti inverzního algoritmu redukuje na problém odhadu kontaktních sil. Byly zde rozvedeny dva možné směry odhadu kontaktních sil, které byly implementovány v prostředí MatlabuFEM a následně vyzkoušeny na jednoduché simulaci. U obou algoritmů se předpokládá silná závislost mezi konvergencí metody a vhodnou volbou počátečních odhadů sil. Tato otázka by měla být předmětem dalšího zkoumání.

6 Simulační prostředí MatlabFEM

V poslední kapitole diplomové práce se zaměříme na implementační část práce. Představíme si framework, na jehož rozšíření a vylepšení jsem pracovala. Ukážeme si, co bylo nutné do frameworku doplnit. A na závěr si porovnáme framework s podobných programem zabývajícím se modelováním hyperelastických simulací FeBiem.

6.1 Představení prostředí MatlabFEM

V úvodu práce bylo uvedeno, že existuje matlabovská implementace hyperelastických simulací, kterou vytvořil vedoucí mé diplomové práce ve spolupráci s institutem Inria. Hlavním důvodem vzniku tohoto femu bylo umožnění rychlého prototypování metod řešení hyperelastických simulací s využitím matematických funkcí, jenž nabízí prostředí Matlabu. To následně umožňuje perspektivní metody implementovat v prostředí SOFA [1].

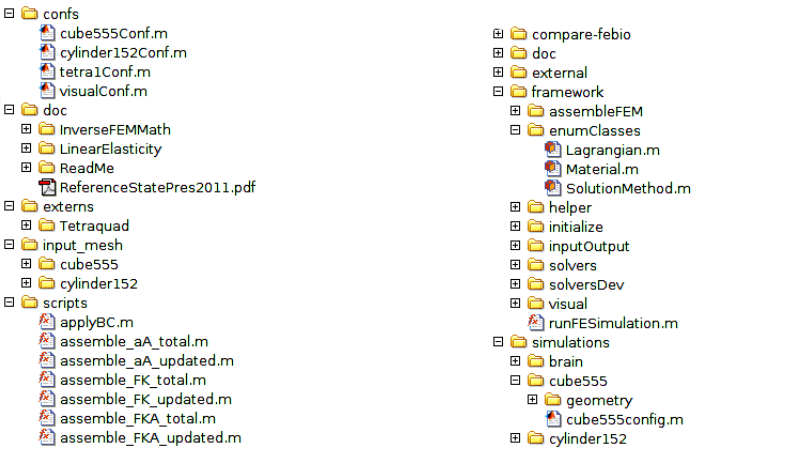
Jelikož se ukázalo, že MatlabFEM může být užitečný nejen jako prototypovací nástroj, ale také jako dobrá výuková pomůcka pro studenty, jejichž programovací znalosti končí na úrovni znalosti Matlabu, začalo se uvažovat o zpřístupnění celého kódu veřejnosti. A právě z tohoto důvodu bylo nutné v kódu udělat změny, které by udělaly program názornější a uživatelsky přívětivější.

6.1.1 Struktura programu

Struktura programu prošla během vývoje programu podstatnými změnami. S vedoucím práce jsme se dohodli na nové struktuře frameworku, která má pomoci v lepší orientaci v frameworku. Změnu adresářové struktury můžeme vidět na obrázku 6.1.

Obrázek 6.1a představuje původní adresářovou, kdy byl kód frameworku rozdělen do pěti hlavních složek - **conf**, **doc**, **externs**, **input_mesh** a **scripts**. Složka **confs** obsahovala konfigurační soubory

1. SOFA (Simulation open framework architecture) je open-oure framework zabývající se medicínskými simulacemi v reálném čase. Více informací a Sofě najdete na jejich webových stránkách https://www.sofa-framework.org.



(a) před restrukturalizací (b) po restrukturalizaci

Obrázek 6.1: Struktura frameworku

s údaji o nastavení simulace, v **doc** se ukrývala dokumentace, v **externs** je nacházely informace o prvcích, z nichž jsou simulovaná tělesa složeny. Údaje o geometrii simulovaných těles byly uloženy v složce **input_mesh** a všechna výpočetní logika frameworku byla k nahlédnutí ve složce **scripts**.

Na obrázku 6.1b vidíme novou strukturu frameworku složenou s pěti složek - **compare-febio**, **doc**, **external**, **framework** a **simulations**. Jediná složka, které zůstala nezměněna je složka **doc** s dokumentací. Složka **externs** byla přejmenována na **external**, která lépe vystihuje obsah složky. Konfigurační soubory s nastavením simulací jsou nově umístěny v složce **simulation**, kde jsou rozděleny do složek podle těles, které jsou objektem deformace, tato složka nově také obsahuje geometrii daného tělesa, která byla dříve umístěna ve složce input_mesh. Nová složka **compare-febio** obsahuje skripty a výsledky porovnání prostředí MatlabFEM a simulačního programu FeBIO, které najdete ve kapitole 6.3. Výpočetní jádro frameworku je umístěna ve složce **framework** a jednotlivé funkce jsou rozděleny do složek podle účelu, ke kterému jsou požívány. Význam funkcí v jednotlivých složkách najdeme v tabulce 6.1.

Složka	Obsah složky
assembleFEM	funkce sloužící k sestavení soustavy rovnic
enumClasses	třídy typu enumerated pro zadávání metod výpočtu
helper	pomocné funkce jako např. kontrolní funkce
initialize	iniciační funkce sloužící k nastavení konfigurace, FEMU, výpočetního modelu atd. před spuštění výpočtu
inputOutput	funkce sloužící pro export a import dat
solvers	řešící funkce
solversDev	experimentální řešiče ve fázi vývoje
visual	vizualizační funkce

Tabulka 6.1: Rozdělení funkcí do jednotlivých složek

6.1.2 Konfigurační soubory

Konfigurační soubor, jeden z nejdůležitějších skriptů, v sobě ukrývá nastavení celé simulace a zároveň spouští výpočet celé simulace. Co všechno musí konfigurační soubor obsahovat najde uživatel v dokumentaci programu, která je součástí příloh B.

Soubor je rozdělen do čtyř hlavních částí - záhlaví, parametry simulace, vizualizační parametry a zápatí. V záhlaví konfiguračního souboru se nachází nastavení domovského adresáře pro výpočet simulace a zajištění viditelnosti všech výpočetních funkcí programu. V zápatí najdeme kontrolu parametrů konfigurace a spuštění simulace.

V části konfiguračního souboru mezi záhlavím a zápatím se nachází definice parametrů simulace a vizualizační parametry, jenž jsou nutné pro zobrazení simulace v Matlabu. Tato část prošla největšími změnami, skupina parametrů udávající nastavení metody výpočtu, typu lagrangianu či vlastnosti materiálu byla převedena z řetězce na výčtový typ *(enumerated)*, a to tak, že byly vytvořeny nové třídy výčtového typu **SolutionMethod**, **Material** a **Lagrangian**. Protože Matlab umožňuje výčtovým typům definovat metody, najdeme u každého výčtového typu metodu *toString(obj)*, která převede výčtový typ na řetězec. Tato metoda se využívá při tvorbě souboru s výsledky simulací,

jehož jméno naznačuje typ použité metody, Lagrangianu, materiálu a příčinu deformace.

```
classdef SolutionMethod < uint32
   enumeration
      INR (1),
      IDNR (2),
   end
   methods
      function str = toString(obj)
         switch obj
             case 1
                 str = 'INR';
             case 2
                 str = 'IDNR';
         end
      end
   end
end
```

Obrázek 6.2: Příklad definice výčtového typu Matlab.

Na obrázku 6.2 najdete kód výčtového typu **SolutionMethod**, který označuje, jaká metoda výpočtu byla zvolena. Zatím tento typ obsahuje pouze dvě metody a to Newtonovu-Raphsonovu metodu (používající analytické derivace) a diskrétní Newton-Raphsonovu metodu (využívající diskrétní aproximace derivací). Vidíme zde i definici metody *toString(obj)*.

Výčtová typ **Lagrangian** je definován stejně jako **SolutionMethod**. Třída **Material** se liší pouze v tom, že obsahuje více funkcí, protože u materiálu můžou existovat různé varianty daného materiálu jako např. stlačitelná, nestlačitelná varianta. Z tohoto důvodu obsahuje třída funkce *isIncompressible(obj)* či *isNearlyIncompressible(obj)*, které vracejí true nebo false podle varianty materiálu.

6.1.3 Kontrola a vyhazování výjimek

Každý programátor by měl očekávat nepozornost či zlomyslnost uživatele a tak by každý program měl kontrolovat správnost vložených parametrů minimálně před spuštěním samotného programu. Z tohoto důvodu MatlabFEM obsahuje skript *checkConfig.m* ve složce **initialize**. Tento skript je spouštěn v záhlaví každého konfiguračního souboru a jedná se o skript ke kontrole nastavení všech parametrů simulace v

rámci dané konfigurace. Skript hlídá přítomnost všech povinných parametrů, kontroluje typografickou správnost parametrů a také vhodně doplňuje konfigurační soubor o potřebné parametry. Více o parametrech a dostupných kombinacích v dokumentaci programu. Na následujících částech kódu budou demonstrovány tři typy kontroly v uvedeném skriptu.

```
13    if (~isfield(params, 'materialType'))
14        error('FATAL ERROR: in configuration file is not definited type of material of
              model like parameter params.typeMaterial');
15    end
```

Obrázek 6.3: Kontrola přítomnosti parametru.

Na obrázku 6.3 můžeme vidět klasickou kontrolu povinného parametru *materialType*, který musí být definován, jelikož určuje materiál tělesa. Jedná se o kontrolu jestli je v konfiguračním souboru tento parametr existuje, nejedná se o kontrolu správného nastavení.

```
95    switch(params.solutionMethod)
96        case SolutionMethod.INR
97        case SolutionMethod.IDNR
98            if(~isfield(params,'perturbation'))
99                error('FATAL ERROR: you need parameter perturbation for discrete NR
                      method like params.perturbation.');
100           end
101       otherwise
102           error('FATAL ERROR: Wrong definition of solution method in configuration
                  file, it have to be type SolutionMethod.NR,...');
103   end
```

Obrázek 6.4: Kontrola typografické správnosti parametru.

Obrázek 6.4 ukazuje kontrolu správného nastavení parametru. Jak už víme z předchozí kapitoly *solutionMethod* musí být nastavena jako výčtový typ. Pokud tomu tak není pokračuje běh programu na řádek 101 a 102 a je vyhozena chyba. V tomto kódu můžeme vidět i kontrolu vzájemné závislosti jednotlivých parametrů a to na řádcích 97 - 100. Pokud je totiž metoda nastavena na diskrétní Newtonovu metodu jenž aproximuje derivace pomocí perturbací musí se v konfiguračním souboru objevit nastavení hodnoty perturbace.

```
163     params.doGravity=0;
164     params.doMassLumping=-1;
165     if(isfield(params,'gravity'))
166         if(length(params.gravity) ~= 3)
167             error('FATAL ERROR: Wrong definition of params.gravity. It has to be a 3D
                    vector, typically [0 0 -9.81].');
168         end
169
170         if (isfield(params, 'totalMass'))
171             params.doMassLumping = 1;
172             params.density = 0;
173         end
174         if (isfield(params, 'density') && params.density > 0)
175             params.doMassLumping = 0;
176         end
177         if params.doMassLumping == -1
178             error('FATAL ERROR: gravity requires either totalMass (lumped version) or
                    density (non-lumped version)');
179         end
180         params.doGravity=1;
181     end
```

Obrázek 6.5: Vhodné doplnění parametrů.

Poslední ukázkou kódu v obrázku 6.5 vidíme vhodné nastavení parametrů konfiguračního souboru. Pokud nebude nastaven parametr gravitace přidají se do konfiguračního souboru nové parametry *doGravity* a *doMassLumping* s příslušnými hodnotami.

Většinu kontrolních operací provádí program ještě před spuštěním řešících metod, a to právě zmiňovanou funkcí *checkConfig.m*. Během výpočtu výpočetní logiky programu jsou pak prováděny kontroly z matematického hlediska jako špatná podmíněnost matice pružnosti, příliš deformované elementy či při uvažování kontaktů stav, kdy těleso projde překážkou. I tyto situace jsou ošetřeny a způsobí tak ukončení programu chybou s vhodnou chybovou hláškou. Jak může vypadat pád programu popř. varování programu ukazuje následující obrázek.

```
Error using checkConfig (line 256)
Error: no motion sources defined, nothing to compute.

Error in cyl152config (line 107)
params = checkConfig(params);
```

(a) error

```
Warning: Problem in reading vertices.
> In readVTK (line 38)
  In setupModel>setupGeometry (line 26)
  In setupModel (line 9)
  In runFESimulation (line 9)
  In cyl152config (line 109)
```

(b) warning

Obrázek 6.6: Chybové ukončení programu a varování

Na obrázku 6.6a je ukázky chybové hlášky programu, kde se můžeme dozvědět, že program spadl ve funkci *checkConfig.m*, protože nebyl definován žádný pohyb tělesa. Na obrázku 6.6b najdeme ukázku varování programu, kde nám program oznamuje, že má problém se čtením vrcholů z vtk souboru funkcí *readVTK*. Toto varování bylo zapříčiněno špatným definováním počtu vrcholů tělesa.

6.1.4 Vizualizační funkce

Jedna z velkých předností programu je možnost vizualizace výsledků přímo v matlabovském prostředí, ale díky používání exportu výsledku do vtk souboru i možnost vizualizace simulace v jiných programech, např. Paraview [2].

Nyní se ale zaměřme na vizualizační funkce v prostředí matlabu. Po dopočítání výsledku simulace program uživateli sám nabídne příkazy, které jsou nutné pro spuštění vizualizací. K zobrazení výsledků simulace využívá program dvě funkce: *visualDeformation.m* a *visualLoading.m*. Obě funkce jsou umístěny v adresáři **framework/visual** a při vizualizaci používaní vizualizační parametry definované v konfiguračním souboru simulace.

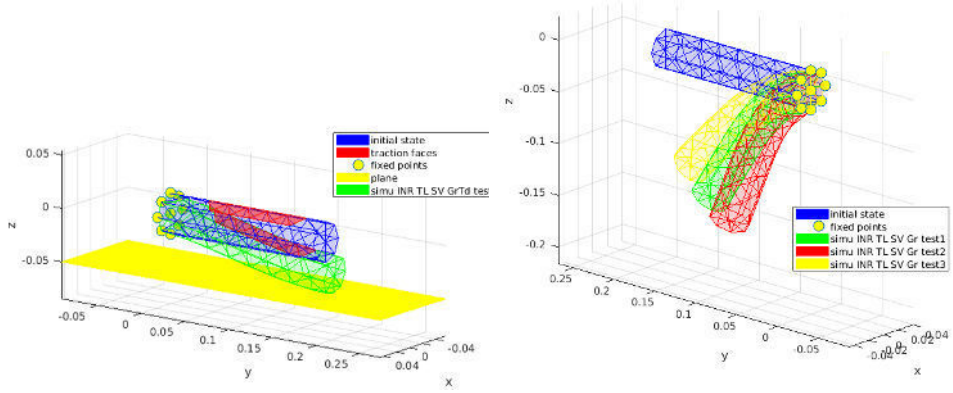
Funkce *visualDeformation* umožňuje uživateli zobrazit počáteční konfiguraci tělesa a konečnou. Jako vstup bere funkce řetězce s jmény souborů, kde jsou uloženy výsledné simulace. Ukázky výstupů funkce najdete v následujících obrázcích.

Na obrázku 6.7a vidíme výsledek použití funkce *visualDeformation* s jednou simulací. Jedná se o zobrazení simulace působení gravitace spolu s povrchovými trakcemi (působí pouze na části válce, které jsou zvýrazněny červenou barvou) na válec s následným kontaktem válce a roviny. Simulace zachycuje pouze dva stavy – modrý válec představující počáteční konfiguraci a zelený válec konečnou.

Obrázek 6.7b zachycuje použití funkce s více simulacemi, vhodnou k porovnání rozdílného chování různých materiálů při stejných deformačních podmínkách. Obrázek ukazuje deformaci válce pomocí gravitace pro různé parametry materiálu. Počáteční pozice, která je pro všechny stejná, je zobrazena modrou barvou, ostatní barvy jsou výsledky simulací.

---

2. http://www.paraview.org/, více o vtk souborech a vizualizaci pomocí paraview v dokumentaci programu



(a) s jedním řetězcem (b) s více řetězci

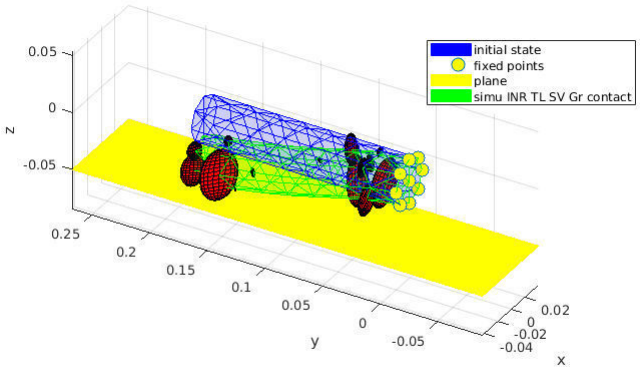
Obrázek 6.7: Ukázky použití skriptu visualDeformation.m.

Druhou vizualizační funkcí je *visualLoading*, která nezobrazuje pouze počáteční a konečnou konfiguraci, ale také průběžné konfigurace vzniklé částečnou aplikací sil, tomuto přístupu se říká kvasi dynamika. Tato funkce umí vizualizovat pouze jednu simulaci.

Ve zdrojových kódech metod mimo matlaboských funkcí najdeme kontrolní funkci *checkVisualParams.m*, která, jak napovídá název, kontroluje vizualizační parametry v případě, že nějaký parametr chybí, upozorní uživatele varovnou hláškou a parametr nastaví na výchozí hodnotu.

Užitečnou vlastnost, kterou mnoho vizualizačních nástrojů zatím neumí a kterou jsme se rozhodli do funkcí dodat v souvislosti s modelování kontaktů, je možnost vizualizace poddajnosti tělesa v bodech. Uživatel si v konfiguračním souboru může nadefinovat, v kterých bodech ho poddajnost zajímá pomocí parametru *complianceNodes*, kde jako vektor zadá správné body. A funkce mu vykreslí k jednotlivých bodům tělesa elipsoidy, které odpovídají poddajnosti daného bodu. Nepovinný parametr, který souvisí s touto funkcí je parametr *scaleCompliance*, kterým může uživatel nastavit měřítko zobrazované poddajnosti.

Na obrázku 6.8 vidíme použití funkce *visualDeformation* s vizualizací poddajnosti. Červené elipsoidy v jednotlivých bodech vymezují velikost poddajnosti pro body tělesa.



Obrázek 6.8: Ukázka vizualizace poddajnosti

6.2 Přidání nových funkcí

Jedním z cílů práce bylo přidání jednoduchých kontaktů mezi tělesem a pevnou překážkou a rovinou. Tohoto cíle bylo dosaženo vytvoření funkce *loadingWithPlaneContact.m* v složce **framework/solvers**. Ta na základě matematických definic kontaktů uvedených v kapitole 3.3 počítá simulaci s uvažování kontaktů. Další funkce, jež byla přidána do stejné složky byla funkce *adaptiveIncrementalLoading.m*, která je aplikací přípustkové metody s adaptivním krokem (popsána v kapitole 2.5.1). Poslední funkce, která byla přidána byla zobrazovací funkce pro poddajnost zmíněná již v předchozí kapitole.

6.2.1 Přidání přípustkové metody s adaptivní krokem výpočtu

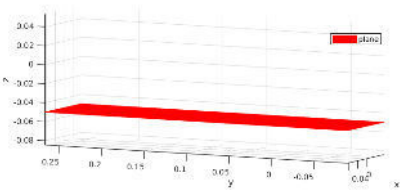
První metodou, která byla přidána do frameworku byla přípustková metoda s adaptivním krokem, jejíž pseudokód najdeme v kapitole 2.5.1. Kód Matlabu je velmi podobný pseudokódu funkce. Malou komplikací při tvorbě funkce v Matlabu bylo rozhodnutí o nepoužívání výjimek, které už Matlab v novějších verzích nabízí. Protože našim cílem je udržet framework dostupný pro co největší spektrum verzí Matlabu místo výjimek, byl použit klasický návratový parametr booleovského typu, který ukazoval, zda při výpočtu Newton-Raphsonovy metody dochází k chybě.

6.2.2 Přidání kontaktů

Pro realizaci kontaktů v prostředí MatlabFEM bylo nutné do konfiguračního souboru přidat nový parametr *params.Plane*, v němž je umožněno uživateli definovat rovinu . Z možných definic roviny, které udává matematiky, byla vybrána definice vycházející z obecné roviny roviny. Uživatel *plane* nastaví jako vektor skalárů udávající parametry obecné rovnice roviny.

```
1  params.plane = [0 0
       1 0.05];
2  params.visuPlane =
     'red';
```

Obrázek 6.9: Zadání parametrů roviny



Obrázek 6.10: Zobrazení roviny

V ukázce kódu 6.9 vidíme zadání parametrů roviny v konfiguračním souboru. První řádek ukazuje přímou definice roviny a řádek číslo 2 nastavení vizualizačních parametrů. Obrázek 6.10 obsahuje vykreslení zadané roviny.

Matematický výpočet simulace s uvažováním roviny je řešen pomocí funkce *loadingWithPlaneContactAdapt.m*, která je volána programem, pokud se v konfiguračním souboru objeví parametr roviny. Jedná se o kombinaci přípustkové metody s adaptivním krokem výpočtu a algoritmu pro řešení kontaktů založeným na LCP (více v kapitole 3.3). Pro funkčnost algoritmu byla také implementována Gauss-Seidelova metoda s projekcí, kterou najdeme ve složce **framework/-solvers** jako funkci *projectiveGaussSeidel.m*.

6.2.3 Přidání zobrazení poddajnosti

Poddajnost tělesa, jak si můžeme přečíst v kapitole 3.1.2, je vlastnost tělesa úzce související s tuhostí a kontakty těles. Vizualizace poddajnosti byla zmíněna již v kapitole 6.1.4. Podmínkou funkčnosti je existence parametru *complianceNodes* v konfiguračním souboru (volitelným parametrem je *scaleCompliance*, který umožňuje upravovat

měřítko zobrazených elipsoidů), která zajistí spuštění exportu poddajnosti pro zvolené body do souboru díky funkci *exportCompliance.m* v složce **framework/inputOutput**.

```
1  function configuration = exportCompliance(configuration, invA,incrementalRatio)
2      for i = configuration.complianceNodes
3          [U, E] = eigs(invA(3*i-2:3*i,3*i-2:3*i));
4          compliance(3*i-2:3*i,:) = [U, E];
5      end
6
7      if incrementalRatio == 1
8          save('complianceFinal.mat','compliance');
9      end
10     save(sprintf('compliance%02d.mat',configuration.l1index),'compliance');
11 end
```

Obrázek 6.11: Kód funkce exportCompliance.m

Obrázek 6.11 ukazuje zdrojový kód funkce *exportCompliance*, která jako vstupní parametry potřebuje strukturu configuration, kde je uloženo, pro jaké body se má poddajnost exportovat, inverzní matici matice tuhosti a incrementalRatio, které je ukazatelem exportu konečné hodnoty poddajnosti. Z kódu funkce vidíme, že nejprve dochází k exportu vlastních čísel a vektorů částí inverzní matice tuhosti odpovídajícím vybraným bodům, poté jsou vlastní vektory a čísla uloženy do matice (kód na řádcích 2 - 5). Takto vytvořená matice je uložena jako matlabovský soubor *complianceXX.mat*, kde XX je nahrazeno číselnou hodnotou indexu výpočtu a slovem final pokud se jedná o konečnou konfiguraci (řádky 7 - 10).

Samotná vizualizace pomocí elipsoidů, kterou můžeme vidět na obrázku 6.8 je prováděna uvnitř vizualizačních funkcí načtením souboru *complianceXX.mat* a voláním funkce *visualizeEllipsoids.m*.

Z kódu funkce *visualizeEllipsoid.m* 6.12 je vidět, že vykreslování probíhá po jednotlivých bodech tak, že je nejprve vytvořen jednotkový elipsoid, který je následně pomocí vlastních vektorů a vlastního čísla příslušného bodu transformován, aby vyjadřoval hodnotu poddajnosti (řádky 2 - 8). Následně jsou hodnoty přeškálovány pomocí měřítka scale a vykresleny matlabovskou funkcí patch (řádky 9 - 10).

```
function visualizeEllipsoid(node, eigValue, eigVectors, scale)
    [x, y, z] = ellipsoid(0,0,0,1, 1, 1);
    for i = 1:size(x,1)*size(x,2)
        vysl = eigVectors*eigValue*[x(i), y(i), z(i)]';
        x(i) = vysl(1);
        y(i) = vysl(2);
        z(i) = vysl(3);
    end
    fvc = surf2patch(scale*x + node(1),scale*y + node(2),scale*z + node(3));
    patch('Faces', fvc.faces, 'Vertices', fvc.vertices, 'FaceColor', [1, 0, 0]);
end
```

Obrázek 6.12: Kód funkce visualizeEllipsoid.m

6.3 Porovnání simulačních programů MatlabFEM a FeBIO

V této kapitole se zaměříme na validaci simulačního prostředí MatlabFEM. Pro účely validace jsme použili simulační prostředí FeBIO. Jedná se o pokročilý software implementující metodu konečných prvků na široké spektrum elastických materiálů. Software má širokou komunitu uživatelů v oblasti biomechaniky a biofyziky [3]. Tento software byl jako validační vybrán z několika důvodů: jedná se o ustálený a dobře zdokumentovaný open-source kód, který byl v minulosti verifikován vzhledem ke standardům používaným v oblasti modelovaní deformací a strukturální analýzy [20]. Navíc umožňuje práci se soubory VTK, které používáme pro zadávání geometrii v prostředí MatlabFEM.

6.3.1 Srovnání deformací a materiálů

Nejprve si musíme ujasnit, které deformace a materiály mají oba softwary společné. Ty pak můžeme použít při srovnávání. Z dokumentace programu B, Matlab FEM Mathematic Backround (součástí dokumentace v **doc/linearElasticity/**) a Febio 2.4 Theory Manual, jež je dostupný na `http://febio.org/febio/febio-documentation`, vidíme, že oba programy používají následující deformace a materiály.

V tabulce 6.2 vidíme, že oba softwary obsahují definice St. Venant - Kirchhoffova materiálu, Mooney-Rivlinova materiálu s logaritmickou

3. http://febio.org

POROVNÁNÍ	MatlabFEM	Febio
MATERIÁL		
St. Venant - Kirchhoff	SV	Isotropic Elastic
Mooney-Rivlin	MRNILOG	Mooney Rivlin
DEFORMACE		
gravitace	gravity with density	Body Loads
síly aplikované v bodech	directionalNodalForc	Nodal Force
tažné síly	directionalTractions	Faces Traction

Tabulka 6.2: Porovnání materiálů a deformací

funkcí a těleso můžou deformovat pomocí gravitace, sil aplikovaných v bodech a tažné síly (trakce).

Složka **compare-febio** obsahuje konfigurační soubory obou programů, použitých při porovnání. K porovnání byly použity dvě sítě (cube555 a cylinder152), které jsou také použity jako příklady simulací v prostředí MatlabFEM. Ve složce **Matlab** najdeme konfigurační soubory pro všechny simulace a složky s výsledky simulací. Ty stejné soubory a složky najdeme ve složce **febio**.

6.3.2 Metody porovnání

Při porovnání obou softwarů bylo využito exportu výsledků do VTK souborů. Ty pak byly porovnávány pomocí vytvořené matlabovské funkce *compareFebioMatlabVTK.m*, kterou najdete v složce **compare-defio**, která se zaměřuje na porovnání geometrie a fyziky.

Porovnání geometrie

Porovnání geometrie bylo provedeno na základě euklidovské vzdálenosti každého bodu tělesa. Euklidovská vzdálenost 2 bodů je definována jako:

$$d_e(A,B) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2} \qquad (6.1)$$

kde $A = [x_a, y_a, z_a], B = [x_b, y_b, z_b]$ jsou body a $d_e(A, B)$ je euklidovská vzdálenost těchto bodů.

Tato vzdálenost by v ideálním případě měla být pro všechny body 0. Protože počítač a softwary mají pouze konečnou přesnost, tak se zde může objevit malá chyba způsobena zaokrouhlováním. Pokud spočítáme euklidovskou vzdálenost všech shodujících se bodů, můžeme při porovnání použít statistiky jako např. maximum, minimum, průměr nebo medián, abychom potvrdili nebo vyvrátili podobnost obou konfigurací.

Porovnání fyziky

Při porovnání fyziky využijeme kritérium s názvem von Misesovo napětí (*von Mises stress*) definovaný pro každý element. Toto napětí je počítáno z Cauchyho tensoru mechanického napětí a výsledkem kritéria je převod tensoru napětí na skalární veličinu podle následujícího vzorce: [21]

$$\sigma_{vms} = \sqrt{\frac{1}{2}\left[(\sigma_{11}-\sigma_{22})^2 + (\sigma_{22}-\sigma_{33})^2 + (\sigma_{33}-\sigma_{11})^2 + 6(\sigma_{12}^2+\sigma_{23}^2+\sigma_{31}^2)\right]} \tag{6.2}$$

kde σ_{vms} je von Misesovo napětí a σ_{xy} vyjadřují složky Cauchyho tensoru mechanického napětí na pozici x, y.

Protože von Misesovo napětí je skalár, můžeme ho jednoduše porovnávat a v každém bodu si vyjádřit rozdíl mezi hodnotami pro jednotlivá prostředí (vyjádřenou v procentech vůči von Misesovu napětí spočítanému prostředím FeBIO):

$$d_{mvs} = \frac{|\sigma_{vms}(F) - \sigma_{vms}(M)|}{\sigma_{vms}(F)} \tag{6.3}$$

kde $\sigma_{vms}(F), \sigma_{vms}(M)$ jsou hodnoty von Mises stresu pro bod F, M, kde F a M jsou odpovídající si body z geometrie F pro FeBIO a M jako MatlabFEM.

V ideálním případě by měla hodnota d_{mvs} všech bodů být 0, ale protože počítač a softwary mají pouze konečnou přesnost, tak se zde může objevit malá chyba způsobena zaokrouhlováním stejně jako v

případě geometrie. I při tomto kritériu můžeme využít statistických funkcí nabízených Matlabem, abychom ověřili podobnost obou konfigurací.

6.3.3 Výsledky porovnání

Porovnání bylo provedeno na základě deformací prováděných nad dvěma tělesy: cube555 a cylinder152; čísla v názvu geometrie určují počet elementů v dané geometrii. Při simulacích byly použity parametry definované v kapitole 6.3.1, výsledky porovnávacích metod pro obě tělesa obsahuje následující text.

Cube555

Porovnání bylo prováděno s parametry, jejichž hodnoty ukazuje tabulka 6.3.

NASTAVENÍ	**hodnoty parametrů**
MATERIÁL	
St. Venant - Kirchhoff	youngsModulus= 30000; poissonsRatio=0.34
DEFORMACE	
gravitace	[0 -9.81 0]; density = 1000
síly aplikované v bodech	[0.1 0.1 0]
tažné síly	[5000 2000 2000]; tractionFaces-febio.csv

Tabulka 6.3: Nastavení parametrů pro Cube555

Výsledky porovnání jsou zobrazeny v tabulce 6.4. Z této tabulky vidíme, že rozdíly v geometrii jsou u poloviny bodů nulové a u druhé poloviny bodů řádově mezi 10^{-9} a 10^{-7}, což je zanedbatelná zaokrouhlovací chyba. Při pohledu na porovnání fyziky jsou rozdíly mezi jednotlivými hodnotami taky zanedbatelné. Pohybují se řádově mezi $10^{-4} - 10^{-5}$% z hodnoty von Misesova napětí spočítané pomocí programu FeBIO, opět zanedbatelná zaokrouhlovací chyba.

POROVNÁNÍ	medián	průměr	maximum
GEOMETRIE			
Gravitace	0	$2.362e^{-9}$	$1e^{-7}$
Síly aplikované v bodech	0	$1.484e^{-8}$	$1e^{-6}$
Tažné síly	0	$2.573e^{-9}$	$1.414e^{-7}$
FYZIKA			
Gravitace	$5.414e^{-5}$%	$7.751e^{-5}$%	$4.760e^{-4}$%
Síly aplikované v bodech	$8.763e^{-5}$%	$1.129e^{-4}$%	$4.440e^{-4}$%
Tažné síly	$5.742e^{-5}$%	$7.544e^{-5}$%	$2.838e^{-4}$%

Tabulka 6.4: Porovnání výsledků pro Cube555

Cylinder152

Porovnání bylo prováděno s parametry, jejichž hodnoty ukazuje tabulka 6.5.

NASTAVENÍ	hodnoty parametrů
MATERIÁL	
St. Venant - Kirchhoff	youngsModulus= 27000; poissonsRatio=0.28
DEFORMACE	
gravitace	[0 -9.81 0]; density = 789
síly aplikované v bodech	[0.02 0 0.03]
tažné síly	[100 50 10]; tractionFaces-febio.csv

Tabulka 6.5: Nastavení parametrů pro Cylinder152

Výsledky porovnání jsou zobrazeny v tabulce 6.6. Z této tabulky vidíme, že rozdíly v geometrii jsou opět u poloviny bodů 0 a druhé poloviny bodů řádově můžeme tento rozdíl označit jako zanedbatelnou zaokrouhlovací chybu. I při porovnání fyziky obou konfigurací můžeme rozdíly, které mezi nimi vznikly označit za zaokrouhlovací chybu.

POROVNÁNÍ	**medián**	**průměr**	**maximum**
GEOMETRIE			
Gravitace	0	$2.161e6^{-8}$	$1e^{-7}$
Síly aplikované v bodech	0	$9.771e^{-9}$	$1e^{-7}$
Tažné síly	0	$5.471e^{-9}$	$1e^{-7}$
FYZIKA			
Gravitace	$1.418e^{-4}$%	$2.093e^{-4}$%	$1.058e^{-3}$%
Síly aplikované v bodech	$1.273e^{-4}$%	$1.498e^{-4}$%	$7.165e^{-4}$%
Tažné síly	$6.392e^{-5}$%	$2.024e^{-4}$%	$6.490e^{-3}$%

Tabulka 6.6: Porovnání výsledků pro Cylinder152

Závěrem můžeme konstatovat, že z hlediska geometrie i fyziky v tomto případě počítají oba softwary stejně, což potvrzuje správnost simulačního kódu prostředí MatlabFEM.

7 Závěr

Při tvorbě diplomové práce jsem se blíže seznámila s metodami matematického modelování deformací těles. Pochopila jsem matematické základy těchto modelů a blíže jsem se seznámila s modely, které se používají pro modelování hyperelastických materiálů. Pochopila jsem základní algoritmy modelování kontaktů mezi deformovatelným tělesem a rigidní překážkou a základní algoritmus kompenzace sil, který představuje pokročilou techniku k rekonstrukci klidového stavu tělesa ze znalosti působících sil. Seznámila jsem se s existujícím simulačním prostředím MatlabFEM, který umožňuje simulace deformací hyperelastických těles.

Během tvorby diplomové práce se povedlo modularizovat simulační prostředí MatlabFEMu a doplnit celý framework tak, že jeho používání pro uživatele je jednodušší a přehlednější. Jako součást práce byla vytvořena rozsáhlá dokumentace v angličtině, která popisuje funkce a prostředí celého programu. V dokumentaci je navíc popsána vizualizace výsledků simulací v Paraview, která je možná díky exportu výsledků simulací VTK souborů.

V programu byly opraveny a vylepšeny možnosti zobrazení výsledků simulací pomocí vizualizačních funkcí. Do těchto funkcí byla přidána možnost zobrazení poddajnosti pro vybrané body tělesa.

Byla doprogramována přírůstková metoda řešení nelineárních rovnic s adaptivním krokem. Na jejíž základě a s pomocí Gauss-Seidelovy metody byla přidána nová metoda pro modelování kontaktů deformovatelného tělesa v pevnou (nedeformovatelnou) rovinou.

V rámci práce bylo také prozkoumána možnost přidání kontaktů do již existující metody kompenzace sil. Obtížnost implementace kontaktů do této metody je definována obtížností odhadu kontaktních sil mezi překážkou a deformovaným tělesem. V práci byly nastíněny dva možné přístupy k tomuto odhadu. Oba přístupy nabízí možnost dalšího zkoumání.

Jako část práce bylo provedeno srovnání MatlabFEMu s programem FeBIO, který je respektovaným simulačním kódem v oblasti modelování nelineárních deformací. Ukázalo se, že pro stejné sítě a vstupné fyzikální parametry oba programy dávají numericky ekvivalentní výsledky.

Jako další směr rozvoje této práce by mohlo být doplnění Matlab-FEMu o jiné techniky používané při modelování deformací těles, popř. rozšíření spektra používaných materiálů. Další možností rozšíření by bylo přidání technik umožňující modelování kontaktů mezi dvěma deformovatelnými tělesy nebo hlubšího zkoumání odhadu kontaktních sil v kombinaci s algoritmem kompenzace sil.

Bibliografie

[1] J.M. Aitchison a M.W. Poole. „A numerical algorithm for the solution of Signorini problems“. In: *Journal of Computational and Applied Mathematics* 94.1 (1998), s. 55–67. ISSN: 0377-0427.

[2] F. Baumgart. „Stiffness — an unknown world of mechanical science?“ In: *Injury* 31 (2000), s. 14–84. ISSN: 0020-1383.

[3] Ted Belytschko, W.K. Liu a B. Moran. *Nonlinear finite elements for continua and structures*. Wiley, 2000. Kap. 1, s. 1–22. ISBN: 9780471987734.

[4] Alexandre Bilger. „Patient-specific biomechanical simulation for deep brain stimulation“. Theses. Université des Sciences et Technologie de Lille, pros. 2014.

[5] Rebecca M. Brannon. „A review of useful theorems involving proper orthogonal matrices referenced to three-dimensional physical space.“ In: s. 13–14. URL: `http://www.mech.utah.edu/~brannon/public/rotation.pdf` (cit. 15. 03. 2017).

[6] Justin D. Bric et al. „Current state of virtual reality simulation in robotic surgery training: a review“. In: *Surgical Endoscopy* 30.6 (2016), s. 2169–2178. ISSN: 1432-2218.

[7] X Ding a JM Selig. „On the compliance of coiled springs“. In: *INTERNATIONAL JOURNAL OF MECHANICAL SCIENCES* 46.5 (2004), s. 703–727. ISSN: 00207403.

[8] Miroslava Dubcová. „Sbírka příkladů z MATEMATIKY II“. In: 1. vyd. Praha: VŠCHT Praha, 2008, s. 82. ISBN: 978-80-7080-706-4.

[9] C. Duriez, C. Andriot a A. Kheddar. „Signorini's contact model for deformable objects in haptic simulations“. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Sv. 4. Zář. 2004, 3232–3237 vol.4.

[10] Christian Duriez et al. „Realistic Haptic Rendering of Interacting Deformable Objects in Virtual Environments“. In: *IEEE Transactions on Visualization and Computer Graphics* 2006.1 (2006), s. 36–47.

[11] Victor D Fachinotti, Alberto Cardona a Philippe Jetteur. „Finite element modelling of inverse design problems in large deformations anisotropic hyperelasticity“. In: *International Journal for Numerical Methods in Engineering* 74.6 (2008), s. 894–910.

[12] Sanjay Govindjee a Paul A Mihalic. „Computational methods for inverse finite elastostatics“. In: *Computer Methods in Applied Mechanics and Engineering* 136.1 (1996), s. 47–57.

[13] David Halliday, Robert Resnick a Jearl Walker. *Fyzika*. 1. Brno, Praha: Vutium, Prometheus, 2001, s. 342–346. ISBN: 80-214-1868-0.

[14] Nazim Haouchine et al. „Impact of Soft Tissue Heterogeneity on Augmented Reality for Liver Surgery“. In: *IEEE Transactions on Visualization and Computer Graphics* 21.5 (2015), s. 584–597.

[15] Shuhaiber JH. „Augmented reality in surgery“. In: *Archives of Surgery* 139.2 (2004), s. 170–174.

[16] C. T. Kelley. *Solving nonlinear equations with Newton's method.* Fundamentals of algorithms. Philadelphia : Society for Industrial a Applied Mathematics, c2003., 2003. ISBN: 0-89871-546-6.

[17] Piaras Kelly. „Solid Mechanics Part III.“ Solid Mechanics Lecture Notes. 2013.

[18] R. M. Kenedi et al. „Tissue Mechanics“. In: *PHYS. MED. BIOL.,* 20.5 (1975), s. 699–717.

[19] Nam-Ho Kim. *Introduction to nonlinear finite element analysis.* Springer, New York, 2015. ISBN: 978-1-4419-1745-4.

[20] Steve A Maas et al. „A comparison of FEBio, ABAQUS, and NIKE3D. Results for a suite of verification problems“. Dis. University of Utah, 2009.

[21] Bob McGinty. *Von Mises Stress*. URL: `http://www.continuum-mechanics.org/vonmisesstress.html` (cit. 10. 04. 2017).

[22] José Luis Morales, Jorge Nocedal a Mikhail Smelyanskiy. „An algorithm for the fast solution of symmetric linear complementarity problems“. In: *Numerische Mathematik* 111.2 (2008), s. 251–266.

[23] K.G. Murty. *Linear Complementarity, Linear and Non Linear Programming*. Sigma series in applied mathematics. Heldermann Verlag, 1988. Kap. 1, s. 1–58. ISBN: 9783885384038.

[24] I. Peterlík, C. Duriez a S. Cotin. „Modeling and real-time simulation of a vascularized liver tissue.“ In: *Medical image computing and computer-assisted intervention : MICCAI ... International Conference on Medical Image Computing and Computer-Assisted Intervention* 15.Pt 1 (2012), s. 50–57.

[25] Igor Peterlík. *Haptic Interaction with Non-linear Deformable Objects*. dizertační. Brno, Czech Republic, 2009.

[26] P.Volný P.Kreml J.Vlček, J.Krček a J.Poláček. „Vázané extrémy“. In: *Matematika II*. 1. vyd. Ostrava: VŠB - Technická univerzita Ostrava, 2007, s. 308–317. ISBN: 978-80-248-1316-5.

[27] „Review of Continuum Mechanics: Kinematics“. Nonlinear Finite Element Methods (ASEN 6107) Course Material. 2006.

[28] M. Sellier. „An iterative method for the inverse elasto-static problem“. In: *Journal of Fluids and Structures* 27.8 (2011), s. 1461–1470. ISSN: 0889-9746. (Cit. 20. 04. 2017).

[29] Eftychios Sifakis a Jernej Barbic. „FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction“. In: *ACM SIGGRAPH 2012 Courses*. SIGGRAPH '12. Los Angeles, California: ACM, 2012, s. 5–13. ISBN: 978-1-4503-1678-1.

[30] Petr Přikryl a Marek Brandner Stanislav Míka. „Speciální numerické metody: numerické metody řešení okrajových úloh pro diferenciální rovnice“. In: *Speciální numerické metody. Numerické metody řešení okrajových úloh pro diferenciální rovnice*. 1. vyd. Plzeň: Vydavatelský servis, 2006, s. 109–130. ISBN: 80-86843-13-0.

[31] Alvaro Valencia, Benjamin Blas a Jaime H Ortega. „Modeling of brain shift phenomenon for different craniotomies and solid models“. In: *Journal of Applied Mathematics* 2012 (2012).

[32] Peter Weinstock. *Lifelike simulations that make real-life surgery safer*. TEDxNatick. 2016. URL: `https://www.ted.com/talks/peter_weinstock_lifelike_simulations_that_make_real_life_surgery_safer#t-1004638` (cit. 17. 04. 2017).

[33] Peter Wriggers. *Nonlinear finite element methods.* Berlin ; Heidelberg : Springer, c2008., 2008. ISBN: 978-3-540-71000-4.

[34] Aneta Zachová. *Studenti medicíny mohou v Brně operovat nanečisto*. Lis. 2013. URL: `http://www.munimedia.cz/prispevek/studenti-mediciny-mohou-v-brne-operovat-nanecisto-6246/` (cit. 17. 04. 2017).

A Definice vybraných materiálů

V této části se zaměříme na definici asi nejznámějších a nejpoužívanějších materiálů. A to materiálů St. Venant - Kirchhoff a Mooney - Rivlin.

A.1 St. Venant - Kirchhoffův materiál

Je nejjednodušší hyperelastický materiál. Existují dvě verze tohoto materiálu - stlačitelný a nestlačitelný. Tento materiál má mechanické napětí definované jako 2. Piola–Kirchhoffův tensor mechanického napětí a deformaci jako Lagrangian-Greenovu deformaci. Jeho hustota energie je definována takto:

Stlačitelná varianta materiálu:

$$W(E) = \frac{\lambda}{2}[tr(E)]^2 + \mu tr(E^2) \tag{A.1}$$

Nestlačitelná varianta materiálu:

$$W(E) = \mu tr(E^2) \tag{A.2}$$

kde λ, μ jsou Lamého koeficienty, E je Lagrangian-Greenův tensor deformace.

Konstituční rovnice pro St. Venant - Kirchhoffův materiál vypadá takto:

$$S = \lambda tr(E)1 + 2\mu E \tag{A.3}$$

kde λ, μ jsou Lamého koeficienty, E je Lagrangian-Greenův tensor deformace, S je 2. Piola–Kirchhoffův tensor mechanického napětí a 1 je jednotková matice.

A.2 Mooney-Rivlinův materiál

Druhý nejvíce používaný materiál je materiál Mooney - Rivlin. Opět se jedná o hyperelastický materiál a existuje mnoho verzí.

Stlačitelný Mooney - Rivlin materiál

Základní typ materiálu a jeho hustota energie je definována takto:

$$W = C_1(\bar{I}_1 - 3) + C_2(\bar{I}_2 - 3) \tag{A.4}$$

$$\bar{I}_1 = J^{-2/3} I_1; I_1 = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 \tag{A.5}$$

$$\bar{I}_2 = J^{-4/3} I_1; I_1 = \lambda_1^2\lambda_2^2 + \lambda_2^2\lambda_3^2 + \lambda_3^2\lambda_1^2 \tag{A.6}$$

$$J = det(F) = \lambda_1^2\lambda_2^2\lambda_3^2 \tag{A.7}$$

kde C_1 a C_2 jsou materiálové konstanty, I_1 a I_2 jsou první a druhý invariant Cauchy - Greenova deformačního tensoru a F je deformační gradient.

Konstituční rovnici pro něj můžeme definovat jako:

$$\sigma = \frac{2}{J}\left[\frac{1}{J^{2/3}}(C_1 + \bar{I}_1 C_2) - \frac{1}{J^{4/3}} C_2 B \cdot B\right] + \left[2D_1(J-1) - \frac{2}{3J}(C_1\bar{I}_1 + 2C_2\bar{I}_2)\right] 1 \tag{A.8}$$

kde σ je Cauchyho mechanické napětí J je determinant deformačního gradientu, C_1 a C_2 jsou materiálové konstanty, I_1 a I_2 jsou první a druhý invariant Cauchy - Greenova deformačního tensoru, B je levý Cauchy-Green deformační tensor a 1 je jednotková matice.

Nestlačitelný Mooney-Rivlin

Pro nestlačitelný Mooney-Rivlin materiál platí, že $\mid J \mid = 1$ takže předchozí hustota energie můžeme definovat jako:

$$W = C_1(\lambda_1^2 + \lambda_2^2 + \lambda_3^2 - 3) + C_2(\lambda_1^2\lambda_2^2 + \lambda_2^2\lambda_3^2 + \lambda_3^2\lambda_1^2 - 3) \tag{A.9}$$

Konstituční rovnice pak vypadá takto:

$$\sigma = 2\left(C_1 + \bar{I}_1 C_2\right) - 2C_2 B \cdot B - \frac{2}{3}\left(C_1\bar{I}_1 + 2C_2\bar{I}_2\right) 1 \tag{A.10}$$

kde σ je Cauchy mechanické napětí C_1 a C_2 jsou materiálové konstanty, $\bar{I}_1$ a $\bar{I}_2$ jsou první a druhý invariant Cauchy - Greenova deformačního tensoru a jsou definovány stejně jako v A.5 a A.6, B je levý Cauchy-Green deformační tensor a 1 je jednotková matice.

Mooney-Rivlin blížící se nestlačitelnosti

Existuje více způsobů jak tento druh materiálu definovat a to v závislosti na tom jaký typ funkce je v definici použit. A tak může být funkce hustoty energie definována jako:

- lineární funkce

$$W = C_1(\overline{I}_1 - 3) + C_2(\overline{I}_2 - 3) + K(J-1)^2 \tag{A.11}$$

- logaritmická funkce

$$W = C_1(\overline{I}_1 - 3) + C_2(\overline{I}_2 - 3) + \frac{1}{2}K\left(log(J)\right)^2 \tag{A.12}$$

kde C_1 a C_2 jsou materiálové konstanty, $\overline{I}_1$ a $\overline{I}_2$ jsou první a druhý invariant Cauchy - Greenova deformačního tensoru a jsou definovány stejně jako v A.5 a A.6, K je bulk modulus, J je determinant deformačního gradient a 1 je jednotková matice.

Konstituční rovnice pak můžeme napsat takto:

- lineární funkce

$$\sigma = \frac{2}{J}\left[\frac{1}{J^{2/3}}(C_1 + \overline{I}_1 C_2)B - \frac{1}{J^{4/3}}C_2B^2\right] + \left[2K(J-1) - \frac{2}{3J}\left(C_1\overline{I}_1 + 2C_2\overline{I}_2\right)\right]1 \tag{A.13}$$

- logaritmická funkce

$$\sigma = \frac{2}{J}\left[\frac{1}{J^{2/3}}(C_1 + \overline{I}_1 C_2)B - \frac{1}{J^{4/3}}C_2B^2\right] + \left[2K\frac{log(J)}{J} - \frac{2}{3J}\left(C_1\overline{I}_1 + 2C_2\overline{I}_2\right)\right]1 \tag{A.14}$$

kde σ je Cauchy mechanické napětí, C_1 a C_2 jsou materiálové konstanty, $\overline{I}_1$ a $\overline{I}_2$ jsou první a druhý invariant Cauchy - Greenova deformačního tensoru a jsou definovány stejně jako v A.5 a A.6, K je bulk modulus, B je levý Cauchy-Green deformační tensor a J je determinant deformačního gradient.

B Dokumentace a zdrojový kód MatlabFEMu

Součástí práce je elektronická příloha, kterou lze najít v přiloženém ZIP archivu **priloha.zip**. Součástí archivu je složka se zdrojovými kódy simulačního prostředí MatlabFEM [1] s vytvořenou dokumentací *UserManual.pdf* ve složce **doc**

1. Součástí vystaveného kódu není pokročilá metoda kompenzace sil.