Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE

Bc. Petr Bulušek

Vizuálně realistické modelování deformací dynamických objektů

Matematický ústav UK

Vedoucí diplomové práce:	Mgr. Jiří Boldyš, Ph.D.
Studijní program:	Matematika
Studijní obor:	Matematické modelování ve fyzice a technice

Praha 2013

Rád bych poděkoval vedoucímu mé diplomové práce Jiřímu Boldyšovi a konzultantovi Jaroslavu Hronovi za pomoc při její tvorbě. Také bych rád poděkoval rodičům za podporu při studiu.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Praze dne 2.8. 2013 Petr Bulušek

Název práce: Vizuálně realistické modelování deformací dynamických objektů

Autor: Bc. Petr Bulušek

Katedra: Matematický ústav UK

Vedoucí diplomové práce: Mgr. Jiří Boldyš, Ph.D., Ústav teorie informace a automatizace AV ČR

Abstrakt: V předložené práci studujeme metody pro simulování fyziky pevných těles a deformovatelných těles. V první kapitole se dá nalézt řešerše některých přístupů k simulaci pevných těles s důrazem na metodu používanou v open source fyzikálním enginu Bullet. Ve druhé kapitole se dají nalézt nejpoužívanější metody pro simulaci deformací opět s důrazem na fyzikální engine Bullet. Dále je studována možnost, jak redukovat dimenzi rovnic, které vzniknou diskretizací parciálních diferenciálních rovnic elastického tělesa metodou konečných prvků. Redukce je studována na příkladu tělesa tvořeného tyčovými elementy.

Klíčová slova: fyzika pevných těles, deformace, redukce modelu

Title: Visually realistic modeling of dynamic objects deformations

Author: Bc. Petr Bulušek

Department: Mathematical Institute of Charles University

Supervisor: Mgr. Jiří Boldyš, Ph.D., Institute of Information Theory and Automation

Abstract: The present work deals with simulation methods for rigid bodies and deformable bodies. In the first chapter you can find research of some methods for simulation of rigid body physics with emphasis on method used in open source physics engine Bullet. In second chapter you can find methods for simulation of deformable bodies, again with emphasis on Bullet physics engine. In last chapter model order reduction technique is presented. This method enables to reduce system of ordinary differential equations. These equations come for example from applying finite element method to partial differential equations describing motion of elastic body. The technique is studied on bar truss systems.

Keywords: rigid body physics, deformation, model order reduction

Obsah

Úvod

Již více jak dvě desetiletí lidé vyvíjejí metody pro simulace pohybu pevných a deformovatelných těles pro aplikace v počítačové grafice. Zřejmě jedna z prvních stěžejních prací v tomto ohledu je [22]. Toto interdisciplinární odvětví kombinuje počítačovou grafiku, Newtonovskou mechaniku, mechaniku kontinua, numerickou matematiku, diferenciální geometrii, vektorový kalkulus, ale i teorii aproximací a další. Jednotlivé modely a metody se liší hlavně podle jejich aplikace. V potaz se musí brát reprezentace modelu, rozsah fyzikálních parametrů, topologické změny atd. Z aplikací zmiňme počítačové hry, animace a speciální efekty ve filmech, ale i různé simulátory. Na základě těchto aplikací se metody rozdělují na ty, které musí pracovat v reálném čase (počítačové hry, simulátory, ...) a na ty, které mohou fyzikální simulaci spočítat i offline (speciální efekty ve filmech, ...). Jestliže mluvíme o real time fyzikálním enginu v počítačové hře, pak u takového algoritmu je potřeba, aby byl schopen generovat zhruba 60 snímků za sekundu, aby simulace působila plynule a věrohodně. Hráči počítačových her chtějí mít ve hře co nejvíce volnosti a možnost interakce s objekty. Jelikož není předem jasné, co se např. v počítačové hře stane, musí být fyzikální engine dostatečně robustní a stabilní. Další aspekty jako rychlost a přesnost fyzikální simulace jdou proti sobě. Real time přesnosti dosáhneme na úkor různých zjednodušení a nepřesností.

Tento obor je v neustálém vývoji i díky stále rychlejší počítačové technice, která dovoluje použití dříve neupočitatelných algoritmů. Rozvoj grafických karet, paralelních výpočtů a fyzikálních procesních jednotek (což jsou procesory sloužící speciálně k počítání fyzikální simulace) také pobízí k vývoji nových algoritmů. Fungující fyzikální engine je komplexní software, při jehož vytváření naráží člověk na poměrně hodně obtíží, např. kvůli tomu, že jsme schopni pouze vzorkovat spojitou realitu v diskrétních časových okamžicích. U 2D fyziky pevných těles, která se používá v mobilních aplikacích a hrách, se stal standardem fyzikální engine Box2D, který je zcela zdarma. Pro 3D fyziku existuje více komerčních fyzikálních enginů, ale i open source projektů, k jejichž vývoji může přispět každý, a jejichž zdrojový kód je volně k dispozici.

Jak jsme již zmínili, různých metod celkově existuje velké množství. Cíle této práce je rešerše některých existujících metod pro simulace pevných a deformovatelných těles s důrazem na open source knihovnu Bullet. Také se podíváme na metodu zvanou redukce modelu, která se dá použít jak v počítačové grafice, tak v jiných aplikacích.

Kapitola 1

Simulace fyziky tuhých těles

Na setkání vývojářů „Eurographics 2012“ (květen 2012) představili autoři Jan Bender, Kenny Erleben, Jeff Trinkle a Erwin Coumans (autor open source fyzikálního enginu Bullet) jejich obsáhlý state-of-the-art článek s názvem „Interactive Simulation of Rigid Body Dynamics in Computer Graphics“ [5], který shrnuje dosavadní práci a pokroky na poli simulací fyziky tuhých těles, které se používají v počítačové grafice (hry, filmy, simulátory, ...). Obsahuje mimo jiné odvození algebraických diferenciálních rovnic, které popisují pohyb systému tuhých těles s Coulombovým třením. Systém, který používá open source kód Bullet physics `http://bulletphysics.org/` je popsán ve článku „Iterative dynamics with temporal coherence“ [6] od Erina Catto (tvůrce 2D open source enginu Box2D) a používá různá zjednodušení oproti rovnicím popsaným v [5]. Nejdříve shrnu teorii potřebnou k popisu tuhého tělesa a pak popíši mechanismus enginu Bullet, zjednodušení, která se používají a jejich výhody.

Tuhé těleso je idealizovaný objekt, u něhož se během pohybu nemění vzdálenost mezi libovolnými dvěma body. Simulace pohybu tuhých těles je analogická numerickému řešení nelineárních obyčejných diferenciálních rovnic, které se nazývají Newton-Eulerovy rovnice. K těmto rovnicím pak musíme přidat další tři podmínky: nepenetrační (kontaktní) vazby, které zabraňují tělesům prostupování, model tření, který vyžaduje, aby kontaktní síly setrvávaly ve svých kuželech tření a ještě tzv. komplementarita, která spojuje kontaktní vazby a kontaktní síly a vynucuje důležitou vlastnost kontaktních sil, aby přestaly působit, jakmile se tělesa začnou pohybovat od sebe.

Pár slov k detekci kolizí

Ve fyzikálním enginu potřebujeme zajistit, aby sebou tělesa neprostupovala. Proto jednou z nejvýznamějších částí enginu je systém na detekci kolizí. Kolize je detekována, pokud se objemy těles překrývají alespoň v jednom bodě. Můžeme postupovat dvěma způsoby. Můžeme zjišťovat přesný čas kolize tím že např. provedeme časový krok Δt a pokud se tělesa překrývají, provedeme znovu pouze $\frac{1}{2}\Delta t$ časový krok. Metodou takovýchto půlení intervalů teoreticky můžeme libovolně přesně zjistit čas kolize a zajistíme také, že se tělesa nebudou překrývat. Tento způsob je pro interaktivní simulace bohužel moc pomalý a tak se používá tzv. retroaktivní detekce. To znamená, že tělesa necháme se překrýt a potom tuto penetraci napravujeme dodatečně. To s sebou samozřejmě přináší také problémy, již samotné řešení penetrací nebo tunneling problem - malá tělesa s vysokými rych-

lostmi prostoupí překážkou, aniž by byla detekována kolize. Mezi objekty nemusí vznikat jen kolizní body ale kolizní hrany anebo celé kolizní plochy. Takové geometrické objekty se aproximují množinou (nejlépe např. rohových) bodů. Výstup detekčního algoritmu pak je kolizní bod (přesná pozice), normálový vektor kolize (je potřeba rozdělit relativní rychlosti a kolizní síly do normálových a tečných směrů) a hloubka penetrace (je potřeba pro následné řešení penetrace). V praxi se netestují všechny objekty mezi sebou, ale používají se algoritmy urychlující celý proces. Objekty se např. obalí do koulí a testuje se nejdříve, zda-li se překrývají tyto koule. Až pak se detekce kolizí zjemňuje a postupně se hledají přesné kolizní body.

1.1 Popis pohybu tuhého tělesa

Pohyb tuhého tělesa se dá rozdělit na translační (pohyb těžiště celého tělesa) a rotační pohyb (rotace okolo těžiště). Každý z těchto pohybů má 3 stupně volnosti, celkově má tedy tuhé těleso 6 stupňů volnosti.

1.1.1 Popis translačního pohybu

Ukládáme aktuální pozici těžiště $\mathbf{x}$ a aktuální rychlost těžiště $\mathbf{v}$. Tyto dvě veličiny jsou svázány kinematickou diferenciální rovnicí

$$\dot{\mathbf{x}} = \mathbf{v}$$

1.1.2 Výpočet těžiště

Souřadnice těžiště obecného tělesa, které vyplňuje v prostoru objem V ($V \in \mathbb{R}^3$) a má hustotu ρ, vypočítáme vzorcem

$$\mathbf{r}_T = \frac{\int_V \rho \, \mathbf{r} \, dV}{\int_V \rho \, dV}$$

1.1.3 Popis rotačního pohybu

Potřebujeme mít uloženou aktuální orientaci tělesa a aktuální úhlovou rychlost.

1.1.4 Orientace tělesa

Orientace tělesa se dá popsat více způsoby. Mezi následujícími popisy samozřejmě existují převodní vztahy.

Tři skalární úhly
Můžeme zvolit např. Eulerovy úhly.

Rotační matice
Značí se většinou $\mathbf{R}$ a jedná se o tzv. ortogonální matici, která má sice 9 prvků, ale jelikož pro její sloupce platí 6 relací ortogonality, redukují se nezávislé parametry na 3 (počet stupňů volnosti).

Jednotkový kvaternion

Jedná se o „normovaný 4D vektor“, jehož první složka se nazývá skalární. Skalární složka se dá vyjádřit pomocí ostatních třech „vektorových“ složek, tedy opět máme 3 nezávislé stupně volnosti.

$$q = w + \mathbf{i}x + \mathbf{j}y + \mathbf{k}z = (s,\mathbf{v}) \in \mathbb{R}^4$$

Mezi imaginárními jednotkami platí následující vztahy

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1, \qquad \mathbf{ij} = \mathbf{k} = -\mathbf{ji}$$

Sdružený a inverzní kvaternion je definován obdobně jako u komplexních čísel.

$$\bar{q} = w - \mathbf{i}x - \mathbf{j}y - \mathbf{k}z$$
$$q^{-1} = \frac{\bar{q}}{q\bar{q}} = \frac{\bar{q}}{\|q\|^2}$$

Kvaternionové násobení je definováno následovně

$$(s_1,\mathbf{v}_1) * (s_2,\mathbf{v}_2) = (s_1 s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2, s_1\mathbf{v}_2 + s_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2)$$

Jedná se o jistou analogii s 2D případem. Rotace ve 2D má jeden stupeň volnosti a takovou rotaci můžeme popsat jednotkovým komplexním číslem (komplexní číslo otočíme kolem počátku tak, že ho vynásobíme jednotkovým komplexním číslem). 3D rotaci o úhel α kolem osy dané jednotkovým vektorem $\mathbf{u}$ můžeme „popsat“ jednotkovým kvaternionen $q = (cos(\frac{\alpha}{2}), sin(\frac{\alpha}{2})\mathbf{u}) = (s,\mathbf{w})$. Rotace vektoru $\mathbf{v}$ o úhel α kolem osy dané jednotkovým vektorem $\mathbf{u}$ pak docílíme kvaternionovým násobením

$$\mathbf{v}_{rot} = q * \mathbf{v} * q^{-1}$$

Vektor $\mathbf{v}$ se vezme jako kvaternion s nulovou skalární složkou a výsledný kvaternion s nulovou složkou se interpretuje znovu jako vektor. Skládání rotací taktéž odpovídá násobení kvaternionů. V počítačové grafice jsou oblíbené zejména jednotkové kvaterniony v kombinaci s rotačními maticemi. Existují tedy samozřejmě převodní vzorečky mezi kvaterniony a rotačními maticemi, které reprezentují tu samou rotaci. Při integraci rovnice s rotační maticí kvůli numerickým zaokrouhlovacím chybám nově získaná matice již nemusí být nutně matice rotace a tak se po pár krocích musí znovu renormalizovat např. Gramm-Schmidtovým ortonormalizačním postupem. Když ale reprezentujeme rotační matici 4-dimenzionálním jednotkovým kvaternionem q, můžeme ho pak jednoduše renormalizovat jako vektor vzorečkem

$$\hat{q} = \frac{q}{\|q\|}$$

1.2 Pohyb tuhého tělesa bez vazeb

Pohybová rovnice hmotného bodu (jednotlivé částice) vychází z Newtonova pohybového zákona

$$\mathbf{F}(\mathbf{x},\dot{\mathbf{x}},t) = m\mathbf{a}(t) = m\dot{\mathbf{v}}(t) = m\ddot{\mathbf{x}}(t),$$

kde m je její hmotnost, $\mathbf{x}$ aktuální poloha, $\mathbf{v}$ rychlost a $\mathbf{a}$ její zrychlení a t značí čas. $\mathbf{F}(\mathbf{x},\dot{\mathbf{x}},t)$ značí, že síla působící na částici typicky nezávisí jenom na čase, ale i na poloze a rychlosti částice. Pokud se částice např. nachází v nějakém netriviálním fyzikálním poli, které na ni působí. Pro usnadnění zápisu budeme závislost na čase vynechávat. Každá obyčejná diferenciální rovnice vyššího řádu než jedna se dá převést na soustavu rovnic prvního řádu. Numerické řešiče tak stačí optimalizovat právě na soustavy prvního řádu. Newtonův pohybový zákon můžeme převést na soustavu dvou rovnic prvního řádu ve tvaru

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{v}\\ \dot{\mathbf{v}} &= \frac{\mathbf{F}}{m}\end{aligned}$$

Pokud budeme používat místo rychlosti $\mathbf{v}$ hybnost $\mathbf{p} = m\mathbf{v}$ (a předpokládáme časově neměnnou hmotnost tělesa) můžeme druhou rovnici zapsat též ve tvaru

$$\dot{\mathbf{p}} = \mathbf{F}$$

V případě pevného tělesa, u kterého nemůžeme zanedbat rotační pohyb, budou tyto rovnice popisovat pohyb jeho těžiště. Ještě musíme přidat rovnice popisující právě zmíněný rotační pohyb. Rovnice nám říkají, jak je ovlivněna rotační matice R(t) (nebo také matice orientace) tělesa v důsledku působení vnějšího momentu síly $\boldsymbol{\tau}(t)$. Analogie hmotnosti m je tzv. matice setrvačnosti I, analogie rychlosti $\mathbf{v}$ je úhlová rychlost $\boldsymbol{\omega}$ a analogie hybnosti $\mathbf{p}$ je moment hybnosti $\mathbf{L}$. Rovnice vypadají následovně

$$\begin{aligned}\dot{\mathrm{R}} &= \mathrm{Skew}(\boldsymbol{\omega})\mathrm{R}\\ \mathbf{L} &= \mathrm{I}\boldsymbol{\omega}\\ \dot{\mathbf{L}} &= \boldsymbol{\tau},\end{aligned} \tag{1.1}$$

kde Skew($\boldsymbol{\omega}$) znamená vytvoření antisymetrické matice z vektoru $\boldsymbol{\omega}$ ve tvaru

$$\begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & \omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}$$

Vektorový součin $\boldsymbol{\omega}\times$ se tak dá nahradit násobením maticí Skew($\boldsymbol{\omega}$) zleva.

A čemu se rovná $\dot{\mathbf{L}}$?

$$\dot{\mathbf{L}} = \dot{\overline{\mathrm{I}\boldsymbol{\omega}}} = \dot{\mathrm{I}}\boldsymbol{\omega} + \boldsymbol{\omega} \times \mathrm{I}\boldsymbol{\omega}$$

Přesné odvození se dá nalézt např. v [1].

Druhý výraz v této rovnici se nazývá Coriolisova síla a ve fyzikálních enginech se často zanedbává (viz. Bullet physics dále). Zanedbání bude znamenat, že se bude zachovávat úhlová rychlost místo úhlového momentu.

Rovnice (1.1) zapsaná pomocí kvaternionů přejde v rovnici (odvození viz např. [1])

$$\dot{q} = \frac{1}{2}\omega * q,$$

kde se na pravé straně násobí kvaterniony a ω se bere jako kvaternion s nulovou skalární částí.

Násobení kvaternionem můžeme přepsat i na maticové násobení

$$\dot{q} = \frac{1}{2}Q\omega, \tag{1.2}$$

kde matice Q je vytvořena z prvků kvaternionu q, detail viz. [13].

Vektorově můžeme celou soustavu obyčejných diferenciálních rovnic pro jedno těleso zapsat ve tvaru

$$\dot{\mathbf{S}} = \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{q} \\ \dot{\mathbf{p}} \\ \dot{\mathbf{L}} \end{bmatrix} = \begin{bmatrix} m^{-1}\mathbf{p} \\ \omega * q/2 \\ \mathbf{F} \\ \boldsymbol{\tau} \end{bmatrix} = \mathbf{G}(t,\mathbf{S}) \tag{1.3}$$

Tyto rovnice se nazývají Newton-Eulerovy rovnice. Matici rotace R vypočítáme z q. Úhlovou rychlost získáme z předešlých rovnic

$$\boldsymbol{\omega} = \mathrm{I}^{-1}\,\mathbf{L} = \mathrm{R}\,\mathrm{I}_{\text{těleso}}^{-1}\mathrm{R}^T\,\mathbf{L}$$

kde I je matice setrvačnosti v globálních souřadnicích a $\mathrm{I}_{\text{těleso}}$ je matice setrvačnosti v lokálních souřadnicích tělesa, kterou můžeme vypočítat před simulací. Tuto soustavu můžeme řešit pomocí vybraného řešiče obyčejných diferenciálních rovnic a získat tak hodnoty proměnných v novém čase $t + \Delta t$.

1.3 Vazby

Když chceme simulovat pohyb reálných objektů, pak se neobejdeme bez vazeb. Tělesa jsou totiž velmi často svázána nějakými vazbami (otáčející se kolo, tlumič, plachta přichycená v některých bodech) a ty je potřeba zavést do enginu. Uvažujeme dva druhy vazeb, permanentní mechanické a kontaktní vazby. Kontakty můžeme dále rozdělit na kolidující, kde je relativní rychlost kontaktních bodů nenulová, a dotykové, kdy se tělesa jen dotýkají a relativní rychlost jejich kontaktních bodů je nulová (např. navršené krabice na sobě).

1.3.1 Permanentní mechanické vazby

Jsou to vazby popsané systémem skalárních rovnic. Zjednodušeně můžeme napsat vazbu mezi dvěma tělesy jako $C = 0$, kde funkce C může záviset na různých proměnných. V praxi vystačíme s vazbami, které závisejí pouze na pozicích a orientacích dvou těles, mezi kterými je vazba definovaná. Mějme například dvě

koule se středy $\mathbf{x}_1$ a $\mathbf{x}_2$ a s poloměry r_1 a r_2. Pokud chceme, aby byly středy těchto koulí neustále ve vzdálenosti L ($L > r_1 + r_2$) od sebe, můžeme takovou vazbu zapsat skalární rovnicí

$$C(\mathbf{x}_1,\mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\| - L = 0 \tag{1.4}$$

Vazby jsou splněny díky tzv. vazebným silám. Ještě se k nim vrátíme.

1.3.2 Kontaktní vazby

Vazbami se též dají popsat izolované bodové kontakty mezi tělesy, ty se dají popsat jednou skalární nerovností. Obdržíme-li detekovaný kontakt dvou těles s těžišti $\mathbf{x}_1$ a $\mathbf{x}_2$ a kontaktními body c_1 (na tělese s těžištěm $\mathbf{x}_1$), c_2 (na tělese s těžištěm $\mathbf{x}_2$) a kontaktní normálou $\mathbf{n}$, pak můžeme zapsat kontaktní vazbu nerovností

$$C(\mathbf{x}_1,\mathbf{x}_2) = (\mathbf{x}_2 + \mathbf{c}_2 - \mathbf{x}_1 - \mathbf{c}_1) \cdot \mathbf{n} \geq 0, \tag{1.5}$$

která vyjadřuje, že sebou tělesa nesmí prostupovat. Vektory $\mathbf{c}_i$ směřují z těžiště do jednotlivých kontaktních bodů.

Jestliže budeme funkci $C(\mathbf{x}_1,\mathbf{x}_2)$ derivovat podle času, dostaneme po první derivaci relativní rychlost kontaktních bodů ve směru normály a po další derivaci relativní zrychlení ve směru normály. Tyto veličiny potřebujeme ke zjištění typu kontaktního bodu. Jestliže jsme měli nerovnost $C \geq 0$ (1.5), pak i když se derivací obecně nerovnost nezachovává, víme, že musí platit i $\dot{C} \geq 0$ a $\ddot{C} \geq 0$. Jak jsme řekli, tyto veličiny představují relativní rychlosti a zrychlení v kontaktních bodech a potřebujeme, aby byly také větší než nula.

1.3.3 Druhy kontaktních bodů

Předpokládejme, že konvexní polygon A svým vrcholem narazí do strany jiného polygonu B v bodě o souřadnici P v čase t. Rychlost tohoto bodu na tělese A bude $\mathbf{v}_A + \boldsymbol{\omega}_A \times \mathbf{r}_A$ a stejný bod na tělese B bude mít rychlost $\mathbf{v}_B + \boldsymbol{\omega}_B \times \mathbf{r}_B$ ($\mathbf{v}_C$, kde C = A nebo B, je rychlost těžiště tělesa). Kontaktní normála $\mathbf{N}$ bude zřejmě kolmá ke kolizní stěně polygonu B. Abychom mohli určit typ kontaktu, zajímá nás relativní pohyb obou kontaktních bodů ve směru kontaktní normály $\mathbf{N}$, tedy průmět $\mathbf{N} \cdot \mathbf{v}_{\mathrm{rel}} = \mathbf{N} \cdot ((\mathbf{v}_A + \boldsymbol{\omega}_A \times \mathbf{r}_A) - (\mathbf{v}_B + \boldsymbol{\omega}_B \times \mathbf{r}_B))$. Na základě této veličiny můžeme kontaktní body rozdělit na tři druhy:

$$\mathbf{v}_{\mathrm{n}} = \mathbf{N} \cdot \mathbf{v}_{\mathrm{rel}}$$

$$\begin{aligned} \mathbf{v}_{\mathrm{n}} &< 0 \quad \text{Kolizní kontakt} \\ \mathbf{v}_{\mathrm{n}} &= 0 \quad \text{dotykový kontakt} \\ \mathbf{v}_{\mathrm{n}} &> 0 \quad \text{Separace,} \end{aligned}$$

přičemž nás zajímají pouze první dva případy.

U dotykového kontaktu v praxi testujeme podmínku $\|\mathbf{v}_\mathrm{n}\| < \varepsilon$, pro nějaké ε malé. Dotykový kontakt se dá rozdělit dále podle tečné složky rychlosti $\mathbf{v}_t$ na

$$\begin{aligned} \mathbf{v}_\mathrm{t} \neq 0 &\quad \text{klouzající kontakt} \\ \mathbf{v}_\mathrm{t} = 0 &\quad \text{valící se kontakt.} \end{aligned}$$

Penetracím zabraňují kontaktní síly, přesněji pokud uvažujeme tření, tak jejich normálové složky. Pokud uvažujeme kontakty bez tření, pak kontaktní síly jsou pouze vektory ve směru kontaktní normály. Jestliže normálové složky kontaktních sil zapíšeme do vektoru $\mathbf{f}_n$, pak potřebujeme, aby tyto síly odstrkovaly objekty od sebe a tedy

$$\mathbf{f}_n \geq 0$$

ve smyslu $\mathbf{f}_{ni} \geq 0, \forall i$.

Takzvaná podmínka komplementarity říká, že kontaktní síla musí působit pouze během kontaktu, t.j. jestliže $C_i = 0$, pak $\mathbf{f}_{ni} > 0$, pokud objekty v kontaktu nejsou, tedy pokud $C_i > 0$, pak $\mathbf{f}_{ni} = 0$. Alespoň jedna z veličin tedy musí být nulová. Jelikož toto musí platit pro všechny kontakty, můžeme tuto podmínku vyjádřit skalárním součinem

$$\mathbf{C}_n \cdot \mathbf{f}_n = 0,$$

kde $\mathbf{C}_n$ je vektor obsahující všechny kontaktní vzdálenosti.

1.4 Řešení kolizních kontaktů, kolizní odezva

Z hlediska fyzikálního enginu nejdůležitější ale také nejobtížnější část je detekce a simulace kolizí.

Detekce kolizí dodá množinu kontaktních bodů a jejich kolizní jednotkové normálové vektory. Na základě těchto dat můžeme rozhodnout o jaký druh kontaktu se jedná.

1.4.1 Příklad jedné 2D kolize bez tření

Jako příklad uvádím kolizi dvou 2D objektů v jednom kontaktním bodě bez tření. Takovou kolizi můžeme analyticky vyřešit, jestliže spočítáme impulsy síly, kterými na sebe obě tělesa zapůsobí. Chceme vypočítat impuls síly působící mezi tělesy a který zabrání penetraci objektů. Vychází se z následujících poznatků. Vektor impulsu síly působící např. na těleso A bude násobek kolizní normály, t.j. $f\mathbf{N}$ pro neznámý skalár f. Rychlosti před kolizí budeme značit horním indexem -, t.j. $\mathbf{v}^-$ a po kolizi s horním indexem +, t.j. $\mathbf{v}^+$. Rychlosti těžišť těles se během kolize změní následovně:

$$\mathbf{v}_A^+ = \mathbf{v}_A^- + \frac{f\mathbf{N}}{m_A}, \quad \mathbf{v}_B^+ = \mathbf{v}_B^- - \frac{f\mathbf{N}}{m_B}$$

Stejně tak se úhlové rychlosti změní na:

$$\boldsymbol{\omega}_A^+ = \boldsymbol{\omega}_A^- + \mathrm{I}_A^{-1}(\mathbf{r}_A \times f\mathbf{N}), \quad \boldsymbol{\omega}_B^+ = \boldsymbol{\omega}_B^- - \mathrm{I}_B^{-1}(\mathbf{r}_B \times f\mathbf{N}),$$

kde $\mathbf{r}_C$, kde C = A nebo B, je vektor spojující těžiště tělesa a bod kontaktu. Využijeme vztah

$$\mathbf{N} \cdot \mathbf{v}_{\text{rel}}^{+} = -\varepsilon \mathbf{N} \cdot \mathbf{v}_{\text{rel}}^{-},$$

který vyjadřuje empirickou vlastnost, kterou by měla kolize splňovat. Koeficient ε vyjadřuje elastičnost kolize a ztrátu kinetické energie. Pro $\varepsilon = 0$ bude kolize plně inelastická a pro $\varepsilon = 1$ plně elastická. Po algebraických úpravách dostaneme vyjádření pro skalár f, které jsme hledali:

$$f = \frac{-(1+\varepsilon)\left(\mathbf{N} \cdot (\mathbf{v}_A^- - \mathbf{v}_B^-) + \left(\boldsymbol{\omega}_A^- \cdot (\mathbf{r}_A \times \mathbf{N}) - \boldsymbol{\omega}_B^- \cdot (\mathbf{r}_B \times \mathbf{N})\right)\right)}{m_A^{-1} + m_B^{-1} + (\mathbf{r}_A \times \mathbf{N})^T \, \mathrm{I}_A^{-1} \, (\mathbf{r}_A \times \mathbf{N}) + (\mathbf{r}_B \times \mathbf{N})^T \, \mathrm{I}_B^{-1} \, (\mathbf{r}_B \times \mathbf{N})}$$

V [13] se dají nalézt vztahy pro impulsy v kontaktním bodě, pokud uvažujeme i tření. V praxi ale v časovém kroku obecně dochází k více kolizním kontaktům. Stačí si představit hromadu krabic naskládaných na sobě. Na jaký druh úlohy tyto situace vedou se pokusíme nastínit v následujícím textu.

1.5 Řešič vazeb

Kromě numerického integrátoru, který integruje Newton-Eulerovy rovnice (1.3) tedy v enginu musí existovat i řešič vazeb. V enginech se setkáváme s tím, že je řešič vazeb rozdělen na moduly, které řeší zvlášť simultánní kolizní kontakty, dotykové kontakty a zvlášť mechanické a všechny ostatní vazby. Takový přístup je vysvětlen např. v knize [12] a dále ukážeme, že se tyto problémy formulují a řeší jako lineární komplementární úlohy (anglická zkratka LCP). Na druhou stranu se dají všechny druhy vazeb řešit stejným přístupem a jedním řešičem. Takový přístup vysvětluje článek [6]. Tento přístup vede na tzv. smíšenou lineární komplementární úlohu (anglicky MLCP). Ta se liší oproti LCP v tom, že komponenty přípustného řešení nejsou jen omezeny zdola nulou, ale některé musí spadat do nějakého omezeného intervalu. Při studiu různých metod se člověk také setkává s výrazy poziční formulace, rychlostní formulace a formulace ve zrychleních. V práci [13] autor vysvětluje v čem spočívá rozdíl mezi těmito různými metodami.

1. Poziční formulace

Chceme splnit $C = 0$. Jestliže je tato vazba porušena a tělesa ji nesplňují, změníme přímo jejich pozice tak, aby vazbu opět splňovala. Toto se řeší pomocí takzvané projekce a metodu proslavil zejména článek [15]. Přímo se tak operuje s pozicemi těles. Jelikož je tato metoda využívána i v implementaci soft body modulu enginu Bullet, ještě se k ní vrátíme.

2. Rychlostní formulace

Předpokládáme, že $C = 0$ a snažíme se splnit $\dot{C} = 0$. Jestliže bude totiž $\dot{C} = 0$, pak bude $C = 0$ splněno i v dalším časovém kroku. Jestliže $\dot{C} \neq 0$, pak se snažíme změnit rychlosti objektů pomocí impulsů, aby byla tato vazba opět splněna.

3. Formulace ve zrychleních

Předpokládáme, že $C = 0$ a $\dot{C} = 0$ a snažíme se splnit $\ddot{C} = 0$. Potom bude totiž v dalším kroku i $\dot{C} = 0$ a s tím i $C = 0$. Toho docílíme tím, že vypočítáme takzvané vazební síly, které působí na tělesa spolu s externími silami.

1.5.1 Lineární komplementární problém

Různé přístupy k řešení simultánních kolizních bodů mají to společné, že vedou na tzv. komplementární problém a jelikož se často, jako je to i u Bulletu, uvažuje zjednodušený model tření, řeší se tzv. lineární komplementární problém. Dříve byly běžnější řešiče, kde v komplementárním problému vystupovaly jako neznámé zrychlení a síly v kontaktních bodech, nyní se používají spíše řešiče na úrovni rychlostí a impulsů v kontaktních bodech. Následuje definice lineárního komplementárního problému.

Nechť $A \in \mathbb{R}^{n\times n}$. Nechť $b \in \mathbb{R}^n$. Najdi $x,w \in \mathbb{R}^n$ splňující

$$\begin{aligned} &w = Ax - b \\ &x \geq 0, w \geq 0 \quad \text{a} \quad (x,w) = 0, \end{aligned}$$

kde $(\cdot,\cdot)$ rozumíme tradiční skalární součin dvou vektorů.

Poslední z podmínek se dá zapsat i jako

$$(x,Ax - b) = 0$$

Jak uvidíme dále, komplementární proměnné v této úloze tvoří buď již zmíněné vektory zrychlení a sil nebo vektory rychlostí a impulsů v kontaktních bodech.

1.5.2 Zpracování simultánních kolizních bodů

Pokud detekujeme v jednom časovém okamžiku více kolizních bodů, mohlo by se zdát, že bude stačit vyřešit každý kontakt zvlášť pomocí vzorců pro jeden kolizní bod. Takovýto přístup používají tzv. sekvenční řešiče a přináší sebou problémy. Vyřešení jednoho kontaktního bodu totiž může např. způsobit penetraci v jiném bodě. Většinou se tedy provede více iterací sekvenčního řešiče. Jiná možnost je sestavit lineární komplementární úlohu pro soustavu simultánních kolizních bodů a vyřešit všechny impulsy najednou. Čerpám z knihy [12], ve které se dá nalézt odvození tohoto problému ve tvaru minimalizace kvadratické funkce. Lineární komplementární úloha se totiž dá ekvivalentně formulovat jako hledání minima jisté kvadratické funkce za omezujících podmínek. V knize [12] čtenář nalezne i přeformulování této minimalizace na lineární komplementární úlohu v rychlostech a impulsech.

Mějme kolekci tuhých těles a n kolizních bodů P_i, $0 \leq i \leq n$. Nechť $\dot{\mathbf{d}}^-$ je vektor, ve kterém jsou uloženy relativní rychlosti před srážkou a nechť $\dot{\mathbf{d}}^+$ je vektor rychlostí po srážce. Aby nedocházelo k penetracím, potřebujeme $\dot{\mathbf{d}}^+ \geq 0$. Nechť $\mathbf{f}$ je vektor, ve kterém shromažďujeme velikosti impulsivních sil ($\mathbf{f}_i = f_i\mathbf{N}_i$, kde $\mathbf{N}_i$ je kontaktní normála v bodě P_i). Impulsivní síly musí být odpudivé a

proto $\mathbf{f} \geq 0$. V knize je odvozeno, že se rychlosti po srážce dají vypočítat lineárním vztahem $\dot{\mathbf{d}}^+ = \mathbf{A}\mathbf{f} + \dot{\mathbf{d}}^-$, kde $\mathbf{A} \in \mathbb{R}^{n \times n}$. Dále definujeme vektor $\mathbf{b}$ s prvky $\mathbf{b}_i = 0$ pokud $\dot{\mathbf{d}}_i^- \geq 0$ a $\mathbf{b}_i = 2\dot{\mathbf{d}}_i^-$ pokud $\dot{\mathbf{d}}_i^- < 0$. Definujme vektor $\mathbf{c}$ s komponentami $\mathbf{c}_i = \|\dot{\mathbf{d}}_i^-\|$. Úloha nalézt vektor impulsivních sil $\mathbf{f}$ se pak dá formulovat jako úloha nalézt minimum konvexní kvadratické funkce $\|\mathbf{A}\mathbf{f} + \mathbf{b}\|^2$ s vazbami $\mathbf{f} \geq 0$, $\mathbf{A}\mathbf{f} + \mathbf{b} \geq 0$ a $\mathbf{A}\mathbf{f} + \mathbf{b} \leq c$.

1.5.3 Zpracování simultánních dotykových kontaktních bodů

Pokud máme soubor více dotykových bodů, dá se jejich simultánní zpracování vyřešit podobně jako kolizní body popsané výše. Pokud jsou relativní zrychlení v kontaktních bodech záporná, snaží se tělesa proniknout do sebe a proto je potřeba mezi nimi působit kontaktními silami (vektor $\mathbf{g} \geq 0$). Vektor zrychlení se dá vyjádřit lineárním vztahem $\ddot{\mathbf{d}} = \mathbf{A}\mathbf{g} + \mathbf{b}$ (Odvození a přesné vzorečky viz kniha [12]). Potřebujeme nalézt vektory $\ddot{\mathbf{d}}$ a $\mathbf{g}$, které splňují lineární vztah výše a vazby $\ddot{\mathbf{d}} \geq 0$, $\mathbf{g} \geq 0$. Navíc máme komplementární vazbu $\ddot{\mathbf{d}} \cdot \mathbf{g} = 0$. Vypočtené síly působící u dotykových kontaktů jsou v řešiči započteny do celkových sil a momentů sil působících na tělesa.

1.5.4 Rychlostní formulace pohybu s vazbami

Jak jsme již zmínili, dají se všechny tyto vazby (kontaktní, mechanické) řešit stejným přístupem uniformě, což vysvětluje jak autor v [12], tak článek [6].

Tento přístup uvažuje vazby pouze mezi dvěma tělesy, které jsou funkcí pozic a orientací těles. Mějme vazbu (označenou indexem k) mezi dvěma pevnými tělesy s indexy i a j

$$C_k(\mathbf{x}_{i_k}, \mathbf{q}_{i_k}, \mathbf{x}_{j_k}, \mathbf{q}_{j_k}) = 0$$

Soustavu více takových vazeb můžeme zapsat vektorově

$$\mathbf{C}\left(\mathbf{x}(t), \mathbf{q}(t)\right) = 0$$

První derivace této vazby je

$$0 = \dot{\mathbf{C}} = \frac{\partial \mathbf{C}}{\partial \mathbf{x}}\dot{\mathbf{x}} + \frac{\partial \mathbf{C}}{\partial \mathbf{q}}\dot{\mathbf{q}} = \left[\frac{\partial \mathbf{C}}{\partial \mathbf{x}} \frac{\partial \mathbf{C}}{\partial \mathbf{q}}\right] \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{q}} \end{bmatrix} = \left[\frac{\partial \mathbf{C}}{\partial \mathbf{x}} \frac{1}{2} \frac{\partial \mathbf{C}}{\partial \mathbf{q}} Q\right] \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix} = G\mathbf{V},$$

kde matice Q je ze vztahu 1.2. Vektor $\mathbf{V}$ představuje jakousi zobecněnou rychlost, vektor, ve kterém je uložena rychlost těžiště i úhlová rychlost.

Druhá derivace vazby je

$$\begin{aligned} 0 = \ddot{\mathbf{C}} &= \left[\frac{\partial \mathbf{C}}{\partial \mathbf{x}} \frac{1}{2} \frac{\partial \mathbf{C}}{\partial \mathbf{q}} Q\right] \begin{bmatrix} \dot{\mathbf{v}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} + \\ &\left[\mathbf{v}^T \boldsymbol{\omega}^T\right] \begin{bmatrix} \frac{\partial^2 \mathbf{C}}{\partial \mathbf{x}^2} & \frac{1}{2} \frac{\partial^2 \mathbf{C}}{\partial \mathbf{x} \partial \mathbf{q}} Q \\ \frac{1}{2} Q^T \left(\frac{\partial^2 \mathbf{C}}{\partial \mathbf{x} \partial \mathbf{q}}\right)^T & \frac{1}{4} Q^T \left(\frac{\partial^2 \mathbf{C}}{\partial \mathbf{q}^2} - \frac{\partial \mathbf{C}}{\partial \mathbf{q}} \mathbf{q} I\right) Q \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix} \\ &= G\dot{\mathbf{V}} + \mathbf{V}^T H \mathbf{V} \end{aligned} \tag{1.6}$$

Odvození tvaru matice H se dá nalézt v [12]. Newton-Eulerovy pohybové rovnice 1.3 pro soustavu pevných těles s vazbami můžeme zapsat vektorově jako

$$M\dot{\mathbf{V}} = \mathbf{F} + \hat{\mathbf{F}} \tag{1.7}$$

$$G\mathbf{V} = 0, \tag{1.8}$$

kde $M = \mathrm{diag}(m_1 I_1, J_1, \ldots, m_n I_n, J_n)$. $\mathbf{F}$ jsou externí síly a $\hat{\mathbf{F}}$ jsou vazebné síly, které zabraňují porušení vazeb. Z rovnice 1.7 vyjádříme $\dot{\mathbf{V}}$ a dosadíme do 1.6. Dostaneme

$$0 = GM^{-1}\left(\mathbf{F} + \hat{\mathbf{F}} + \mathbf{V}^T H \mathbf{V}\right)$$

Podle principu virtuální práce, který říká, že vazebné síly nekonají práci a nedodávají tak systému s vazbami energii, je vektor vazebných sil $\hat{\mathbf{F}}$ roven

$$\hat{\mathbf{F}} = G^T \boldsymbol{\lambda},$$

Dosazením dostaneme výslednou soustavu pro vektor neznámých koeficientů $\boldsymbol{\lambda}$. Těmto koeficientům se často říká Lagranegovy multiplikátory.

$$GM^{-1}G^T\boldsymbol{\lambda} = -\left(GM^{-1}\mathbf{F} + \mathbf{V}^T H \mathbf{V}\right) \tag{1.9}$$

1.6 Bullet - modul pevných těles

Článek [6] popisuje tento uniformní přístup, který je implementován v kódu Bullet a který se trochu liší od předchozího postupu.

1.6.1 Nerovnostní vazby, kontaktní model

Ukažme si, jak se postupuje u nerovnostních vazeb. Nerovnostní vazby se převádějí na nerovnosti pro multiplikátory $\boldsymbol{\lambda}$, to znamená, že můžeme specifikovat omezený interval $\left[\lambda_i^-, \lambda_i^+\right]$ pro každou komponentu: $\lambda_i^- \leq \lambda_i \leq \lambda_i^+$. V případě, že máme rovnostní vazbu, máme $\lambda_i \in (-\infty, +\infty)$. Ukažme si to na případu kontaktní vazby (1.5)

$$C(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_2 + \mathbf{c}_2 - \mathbf{x}_1 - \mathbf{c}_1) \cdot \mathbf{n} \geq 0,$$

Po kontaktní vazebné síle, která působí ve směru vektoru $\mathbf{n}$ chceme, aby od sebe tělesa odstrkovala (nepřitahovala), takže požadujeme nerovnost $0 \leq \lambda < \infty$. U tření také definujeme omezení na multiplikátory. Obecně se vazby s nějakým limitem promítnou do omezení na Lagrangeovské multiplikátory.

1.6.2 Nenulová rychlostní vazba, Baumgarteho stabilizace

Zmínili jsme se, že při retrospektivní detekci kolizí dochází k penetracím mezi objekty, protože se kolize detekuje až když k ní už došlo. Stejně tak dochází kvůli numerickým chybám k porušení vazeb. Náprava těchto chyb se dá realizovat pomocí nenulové pravé strany u rychlostní vazby a tato metoda je známá též pod názvem Baumgarteho stabilizace [6]. Uvažujeme rychlostní vazbu ve tvaru

$$\dot{C} = G\mathbf{V} = -\beta C = \xi \neq 0,$$

kde pravá strana bude nenulová kvůli chybě ve vazbě $C \neq 0$. Rovnice $\dot{C} + \beta C = 0$ je lineární diferenciální rovnice s řešením $C(t) = C(0)exp(-\beta t)$. Jelikož je při penetraci $C < 0$, potřebujeme $\beta > 0$, aby C šlo k nule. Tuto rovnici řešíme zároveň během simulace. Jestliže máme pevný časový krok, ukazuje se, že je kvůli stabilitě potřeba volit $0 < \beta < 2/\Delta t$, viz [12].

1.6.3 Coulombův izotropický model tření

Coulombův izotropický model tření, který vyžaduje, aby kontaktní síla byla v tzv. kuželu tření, který je popsán rovnicí

$$f_t^2 + f_o^2 \leq \mu^2 f_n^2,$$

kde f_t, f_o jsou tečné složky kontaktní síly, f_n je normálová složka a μ je koeficient tření. Tento model vnáší do formulace rovnic nelinearitu, detaily viz [5].

Proto se používá zjednodušený model tření, který se dá formulovat pomocí dvou vazeb přímo na „úrovni rychlostí“

$$\dot{C_{u1}} = (v_2 + \omega_2 \times r_2 - v_1 - \omega_1 \times r_1) \cdot u1$$
$$\dot{C_{u2}} = (v_2 + \omega_2 \times r_2 - v_1 - \omega_1 \times r_1) \cdot u2,$$

kde $u_1 \times u_2 = n$ jsou tečné vektory kolmé na normálu n v kontakním bodě. Tyto vazby vyjadřují, že se třecí síla má snažit relativní rychlost v kontaktním bodě projektovanou do těchto tečných složek vynulovat. Na Lagrangeovy multiplikátory vztahující se ke třecí vazbě se aplikuje následující omezení

$$-\mu m_c g \leq \lambda_{u1} \leq \mu m_c g$$
$$-\mu m_c g \leq \lambda_{u2} \leq \mu m_c g,$$

kde μ je koeficient tření a m_c je určitá hmotnost přiřazená kontaktu. Odpovídá to tomu, že Coloumbův kužel tření je aproximován krychlí. Velikost tření nezávisí na velikosti normálové složky. Z nakupených kostek na sobě tak například spodní kostky vytáhneme pomocí stejné síly jako vrchní kostky. Také se nerozlišuje mezi statickým a dynamickým třením, uvažujeme pouze jeden koeficient tření μ. V praxi se ukazuje, že je takový model dostačující.

1.6.4 Časová diskretizace

Viděli jsme, že problém vede na řešení soustavy (1.9). [6] navrhuje alternativní postup. Diskretizací rychlosti pomocí konečné diference dostaneme

$$\dot{\mathbf{V}} \approx \frac{\mathbf{V}^2 - \mathbf{V}^1}{\Delta t}$$
$$M(\mathbf{V}^2 - \mathbf{V}^1) = \Delta t(\mathbf{F} + G^T \boldsymbol{\lambda})$$

Pro novou rychlost požadujeme splnění rychlostní vazby

$$G\mathbf{V}^2 = \xi$$

Rovnice můžeme upravit do tvaru

$$GM^{-1}G^T\boldsymbol{\lambda} = \frac{\xi - G\mathbf{V}^1}{\Delta t} - GM^{-1}\mathbf{F}$$

Jestliže řešením získáme $\boldsymbol{\lambda}$, vypočítáme nové rychlosti a pozice aktualizujeme jako

$$\begin{aligned} x^2 &= x^1 + \Delta t v^2 \\ q^2 &= q^1 + \frac{\Delta t}{2}\omega^2 * q^1, \end{aligned}$$

kde v^i a ω^i jsou části vektoru $\mathbf{V}$ náležící k pozicím a orientacím.

Řešíme tzv. smíšenou lineární komplementární úlohu (angl. MLCP), která je tvaru

$$\begin{aligned} \mathbf{V} &= A\boldsymbol{\lambda} - \boldsymbol{\eta} \\ \boldsymbol{\lambda}^- &\leq \boldsymbol{\lambda} \leq \boldsymbol{\lambda}^+ \\ V_i = 0 &\Leftrightarrow \lambda_i^- \leq \lambda_i \leq \lambda_i^+, \forall i \\ V_i \geq 0 &\Leftrightarrow \lambda_i = \lambda^-, \forall i \\ V_i \leq 0 &\Leftrightarrow \lambda_i = \lambda^+, \forall i \end{aligned}$$

Oproti lineární komplementární úloze se liší v tom, že zde máme větší omezení na multiplikátory $\boldsymbol{\lambda}$. Existuje více metod, jak takovou úlohu řešit. Stručné shrnutí se dá nalézt například v [13]. Možnost, kterou navrhuje [6] je tzv. projektivní Gauss-Seidelova metoda. Vezmeme Gauss-Seidelovu metodu na řešení lineárních rovnic a v každé iteraci projektujeme multiplikátory na přípustné intervaly.

Obecně je vhodné v interaktivních enginech používat iterativní řešiče komplementárních úloh, protože pokud máme např. málo času na dopočtení všech iterací, tak můžeme stále dát k dispozici alespoň aproximativní řešení úlohy. Je výhodné za počáteční iteraci v dalším kroku zvolit výsledek získaný v předešlém kroku, neboť často se konfigurace příliš nezmění. Tomuto se říká cachování kontaktů (angl. contact caching). Jelikož i dotykové kontakty řešíme na úrovni rychlostí a impulsů, tak pokud např. narovnáme krabice na sebe, tak po nějaké době takováto struktura spadne, viz [6]. Článek také zmiňuje, že pokud zakomponujeme do algoritmu contact caching, k problémům u navršených objektů přestane docházet a zůstanou stabilní.

1.7 Změkčení vazeb

Velice často potřebujeme simulovat tlumený harmonický oscilátor (odpružení vozidel, ...). Známe rovnici pro harmonický oscilátor i jak ji numericky řešit.

Fyzikální engine jako Bullet má v sobě řešič impulsů a pracuje s vazbami. Jak jsme viděli, pohybové rovnice se integrují semi-implicitní (symplektickou) Eulerovou metodou. Erin Catto ve svém textu [7] vysvětluje, že zavedeme-li tzv. změkčené vazby a řešíme je zmíněnou semi-implicitní Eulerovou metodou, dostaneme stejný výsledek, jako když řešíme harmonický oscilátor implicitní Eulerovou metodou. To znamená, že budeme mít vždy stabilní algoritmus a jak je ukázáno v [7], můžeme příjemně nastavovat konstanty v pružině pomocí frekvence a tlumení. Rovnice harmonického oscilátoru v jedné dimenzi vypadá následovně

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = 0,$$

kde m je hmotnost hmotného bodu, c je konstanta tlumení a k konstanta tuhosti pružiny. Pokud tuto rovnici řešíme implicitní integrační metodou

$$m\frac{v_2 - v_1}{h} + cv_2 + k(x_1 + hv_2) = 0,$$

kde h je zvolený časový krok, dostaneme pro novou rychlost

$$v_2 = \frac{v_1 - \frac{hk}{m}x_1}{1 + \frac{hc}{m} + \frac{h^2k}{m}}$$

Mějme tzv. změkčenou vazbu

$$m\frac{dv}{dt} = \lambda$$
$$v + \frac{\beta}{h}x + \gamma\lambda = 0$$

Pokud tyto rovnice řešíme semi-implicitně

$$m\frac{v_2 - v_1}{h} = \lambda$$
$$v_2 + \frac{\beta}{h}x_1 + \gamma\lambda = 0$$

Z první rovnice vezmeme λ a dosadíme do druhé, dostaneme

$$v_2 = \frac{v_1 - \frac{\beta}{m\gamma}x_1}{1 + \frac{h}{m\gamma}}$$

Když porovnáme koeficienty u x_1 a v_1, dostaneme vztahy

$$\gamma = \frac{1}{c + hk}$$
$$\beta = \frac{hk}{c + hk}$$

Při použití těchto vztahů tedy dostaneme systém vazeb, který zůstane při řešení semi-implicitní metodou vždy stabilní.

Obecně pak každou rychlostní vazbu můžeme „změkčit“ pomocí dvou parametrů

$$G\mathbf{V} + \frac{\beta}{h}C(x) + \gamma\lambda = 0$$

Např. ve fyzikálním enginu Box2D je v definici vzdálenostní vazby mezi dvěma tělesy možnost zároveň definovat frekvenci a konstantu tlumení, v případě, že chceme aby vzdálenostní vazba nebyla tuhá, ale chovala se jako harmonický oscilátor. Obdobné konstanty se dají nalézt i v Bulletu. Ve zmiňovaném článku se dá nalézt vztah mezi konstantami β,γ a frekvencí a konstantou tlumení harmonického oscilátoru.

Kapitola 2

Simulace fyziky deformovatelných těles

Metody pro simulování deformací objektů v počítačové grafice by se daly rozdělit na tyto:

1. Předem spočítané animace
2. Geometrické (nefyzikální)
3. Fyzikální (anglicky physically based)

Článek [19] z roku 2005 obsáhle shrnuje fyzikální metody, které se používají v počítačové grafice jak pro offline, tak i pro real time simulace, které jsou důležité v počítačových hrách. Kvůli různorodosti reálných objektů a cílů, kterých chce člověk v simulaci dosáhnout lidé vyvinuli mnoho metod. Metody by se daly dále rozdělit např. podle těchto kritérií:

1. Dimenze objektu: 1D (vlas), 2D (látka), 3D (kostka)
2. Materiálové vlastnosti: tuhá tělesa (bez deformace), elasticita, plasticita, viskozita, elastoplastoviskozita,...
3. Offline versus real time simulace
4. S tím související přesnost, stabilita, rychlost, vizuální realističnost

Budeme se zabývat fyzikálními metodami a poziční metodou (anglicky position based dynamics), která je implementována v Bullet soft body modulu. Pro svoji jednoduchost je čím dál víc oblíbená.

2.1 Model hmotný bod - pružina

Intuitivní a jednoduchá metoda, jak deformovatelné objekty simulovat, je pomocí modelu hmotný bod - pružina (angl. mass spring model). Nevycházíme z popisu, který dává k dispozici mechanika kontinua, ale diskretizujeme objekt hned od začátku soustavou hmotných bodů spojených pružinkami, či přidanými tlumiči. Tento postup s úspěchem využívá např. engine Rigs of Rods `http://www.rigsofrods.com/`, který simuluje deformace vozidel. Vozidlo je sestaveno

kompletně z velkého množství harmonických oscilátorů. Různé části vozidla mají podle druhu materiálu různě definovány konstanty tuhosti, tlumení a koeficienty řešící plasticitu. Plasticita se dá řešit změnou klidových vzdáleností pružinek, když přesáhneme např. maximální sílu (napětí) v pružince. Vždy se jedná o nějaký spíše heuristický přístup.

Model je složen z N hmotných bodů s hmotnostmi m_i, pozicemi $\mathbf{x}_i$ a rychlostmi $\mathbf{v}_i$, $i \in 1, \ldots, N$. Tyto hmotné body jsou propojeny množinou S pružinek s vlastnostmi (i, j, L_0, k_s, k_d), kde i a j jsou indexy spojených bodů, L_0 je klidová vzdálenost, k_s je konstanta tuhosti a k_d je konstanta tlumení pružinky. Síly, které působí na propojené body jsou

$$\begin{aligned}
\mathbf{f}_i &= \mathbf{f}^s(\mathbf{x}_i,\mathbf{x}_j) + \mathbf{f}^d(\mathbf{x}_i,\mathbf{v}_i,\mathbf{x}_j,\mathbf{v}_j) \\
&= k_s \frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|}(|\mathbf{x}_j - \mathbf{x}_i| - L_0) + k_d(\mathbf{v}_j - \mathbf{v}_i) \cdot \frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|} \\
\mathbf{f}_j &= -\mathbf{f}_i,
\end{aligned}$$

K řešení vzniklých obyčejných diferenciálních rovnic se pak zavolá vhodná integrační metoda. Ve fyzikálním enginu potřebujeme dosáhnout nejlépe nepodmíněné stability, a proto je vhodné použít implicitní Eulerovu metodu. Řešení pak v jednotlivém časovém kroku povede na řešení nelineární soustavy rovnic, protože působící síly jsou nelineární funkce pozic spojených bodů.

2.2 Bullet - modul deformovatelných těles

V tzv. soft body physics modulu obsaženém v Bullet enginu je implementována tzv. poziční dynamika, která je popsána např. ve článku [18]. Definují se vazby, ze kterých není počítána energie a následně síly jako v tzv. penalizační metodě, ale pozice jsou pomocí projekce manipulovány přímo. To sebou přináší mnohé výhody jako např. jednoduchou změnu pozic objektů. Když chceme např. přichytit deformovatelný objekt k pevnému tělesu, tak jednoduše pomocí vazby definujeme, že mají objekty sdílet stejný bod. Na druhou stranu není fyzikálně odůvodnitelný, i když výsledky vypadají velice přijatelně. Pomocí této metody se dají simulovat i pevné látky (jako hmotné body spojené vzdálenostními vazbami), špatně se pak ale řeší kolize a dotykové kontakty.

Dynamický objekt je reprezentovaný N vrcholy a M vazbami. Vrchol $i \in [1, \ldots, N]$ má hmotnost m_i, pozici $\mathbf{x}_i$ a rychlost $\mathbf{v}_i$. Obdobně jako u modelu hmotný bod - pružina. Vazba $j \in [1, \ldots, M]$ má kardinalitu n_j, reprezentující funkci $C_j : \mathbb{R}^{3n_j} \to \mathbb{R}$, množinu indexů $\{i_1, \ldots, i_{n_j}, \}$, $i_k \in [1, \ldots, N]$, tuhostní parametr $k_j \in [0,1]$, který jakýmsi způsobem reprezentuje, jak moc „měkká“ vazba bude a s tím i jako moc deformovatelné bude celé těleso. Dále je definován typ vazby, což je buď rovnost nebo nerovnost. V článku [18] se dají nalézt různé druhy vazeb. Z analýzy kódu Bullet se ale zdá, že je v Bulletu používána pouze vzdálenostní vazba mezi jednotlivými vrcholy, která říká, jak daleko od sebe vrcholy mají zůstat, viz. (1.4).

V časovém kroku se aktualizují síly, vypočítají se nové rychlosti a pozice. Ty můžeme počítat klidně nejjednodušší explicitní Eulerovou metodou, jelikož je pak budeme dodatečně napravovat. Kromě pevných vazeb se vypočítají i kontaktní vazby, které se neustále mění. Pozice se iteračně korigují pomocí takzvané projekce a pak se uloží nové rychlosti a nové pozice do starých.

Projekce vazeb
Při projekci vnitřních vazeb se dbá na to, aby platil zákon zachování hybnosti a momentu hybnosti. Popsaná metoda oba tyto zákony splňuje. Pokud se všechny vazby spojí do vektoru C a pozice vrcholů do vektoru $\mathbf{p}$, pak pokud pozice při korekci posuneme ve směru $\nabla_{\mathbf{p}} C$, potom budou oba zákony splněny. To znamená, že hledáme λ tak, aby pro posunutí $\Delta\mathbf{p}$ platilo

$$\Delta\mathbf{p} = \lambda \nabla_{\mathbf{p}} C(\mathbf{p})$$

V [18] je ukázáno, že pro jednotlivé projekce $\Delta\mathbf{p}_i$ dostáváme vztahy

$$\Delta\mathbf{p}_i = -s w_i \nabla_{\mathbf{p}_i} C(\mathbf{p}_1, \ldots, \mathbf{p}_n),$$

kde

$$s = \frac{C(\mathbf{p}_1, \ldots, \mathbf{p}_n)}{\sum_j w_j \|\nabla_{\mathbf{p}_j} C(\mathbf{p}_1, \ldots, \mathbf{p}_n)\|^2}$$

a $w_j = \frac{1}{m_j}$, kde m_j jsou různé hmotnosti jednotlivých bodů. To znamená, že do projekce zahrnujeme různé hmotnosti, a podle toho je škálujeme.

Projekce $\Delta\mathbf{p}$ se dále násobí výrazem $k' = (1-k)^{1/n_s}$, kde k je tuhostní parametr a n_s je počet iterací projekčního řešiče. Násobíme tímto výrazem, abychom po n_s iteracích dostali lineární závislost na k.

$$\Delta\mathbf{p}(k')^{n_s} = \Delta\mathbf{p}(1-k)$$

Obdržená tuhost materiálu bude závislá na časovém kroku, ale v enginech se většinou používá pevný časový krok a proto toto není problém.

Aspekty poziční dynamiky v Bulletu
V kódu bullet soft body se dají rozpoznat aspekty této poziční dynamiky. Deformovatelný objekt sestává z tzv. „Nodes“ a „Links“ (názvy odpovídají patřičným datovým strukturám), což jsou hmotné body spojené vazbami. Kromě přímých vzdálenostních vazeb, které spojují jednotlivé vrcholy, je v kódu např. funkce generateBendingConstraints(), která vytváří vzdálenostní vazby nejen mezi přímo sousedními vrcholy, docílí se tak toho, že např. objekt, kterým chceme simulovat látku, bude vykazovat odpor proti ohýbání (anglicky bending). V následující funkci se dá rozpoznat eulerovské integrování pozic jednotlivých vrcholů, kdy se nejprve naakumulují síly funkcí applyForces() a pak se přepočtou rychlosti a pozice u všech jednotlivých vrcholů.

```
1 void        btSoftBody::predictMotion(btScalar dt)
2 ...
3 applyForces();
4
5 /* Integrate */
6
7   for(i=0,ni=m_nodes.size();i<ni;++i)
8   {
9     Node& n=m_nodes[i];
```

```
10      n.m_q = n.m_x;
11      n.m_v +=  n.m_f*n.m_im*m_sst.sdt;
12      n.m_x +=  n.m_v*m_sst.sdt;
13      n.m_f = btVector3(0,0,0);
14    }
```

Zřejmě stěžejní metoda celého soft body kódu je metoda s názvem PSolve-Links. V ní se iteruje přes všechny vzdálenostní vazby deformovatelného tělesa, načtou se oba vrcholy, které vazba spojuje (zde pojmenovány a a b). Spočítá se aktuální vzdálenost obou vrcholů a pomocí nastavené klidové vzdálenosti se opraví pozice vrcholů ve směru spojnice. Kód je na řádcích 14-16. V proměnné m_c1 je uložena klidová vzdálenost spojnice na druhou (označme r_0), v proměnné len je aktuální vzdálenost na druhou, po vyřešení kontaktů, integrování atd. (označme r). V proměnné m_c0 je uložena konstanta odpovídající vážení (škálování) jednotlivých vrcholů (podle toho jaké mají hmotnosti) přenásobená ještě faktorem $kLST$-linear stiffness coefficient $\in$ [0,1], který udává jak rychle se mají spojnice opravit a jak moc bude objekt deformovatelný (viz. předchozí text). Na řádku 15 se tak dá rozpoznat vzoreček

$$a- = (b-a) * \frac{r_0^2 - r^2}{r_0^2 + r^2} * \frac{1/m_a}{(1/m_a + 1/m_b)} * kLST$$

Tyto vzorečky téměř odpovídají vzorečkům (10) a (11) v článku [18]. Těm by odpovídal tvar

$$a- = (b-a) * \frac{r_0 - r}{r} * \frac{1/m_a}{(1/m_a + 1/m_b)} * kLST$$

Když tímto výrazem přepíšu existující kód, funguje simulace také, oba lišící se výrazy se totiž chovají v okolí klidové vzdálenosti podobně.

```
1  void  btSoftBody::PSolve_Links(btSoftBody* psb,btScalar kst,btScalar
       ti)
2  {
3    for(int i=0, ni=psb->m_links.size(); i<ni; ++i)
4    {
5      Link& l=psb->m_links[i];
6      if(l.m_c0>0)
7      {
8        Node& a=*l.m_n[0];
9        Node& b=*l.m_n[1];
10       const btVector3 del=b.m_x-a.m_x;
11       const btScalar  len=del.length2();
12       if (l.m_c1+len > SIMD_EPSILON)
13       {
14         const btScalar  k=((l.m_c1-len)/(l.m_c0*(l.m_c1+len)))*kst;
15         a.m_x-=del*(k*a.m_im);
16         b.m_x+=del*(k*b.m_im);
17       }
18     }
19   }
20 }
```

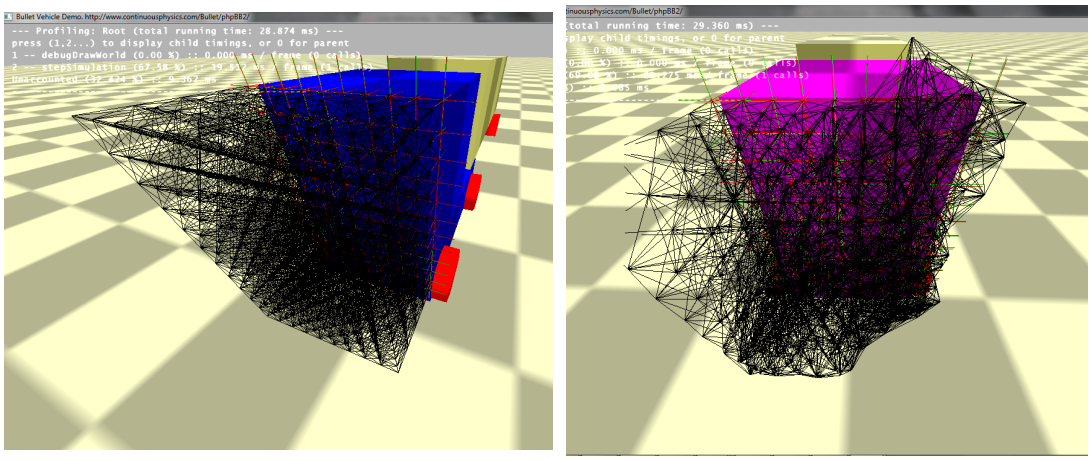
Kód je zapsaný tímto způsobem zřejmě kvůli tomu, aby se ve jmenovateli neobjevila nula a nedošlo k numerickým problémům. Funkce PSolve tedy opravuje spojnice mezi jednotlivými vrcholy tak, aby se zachovávaly jednotlivé vzdálenosti.

Jednoduchá plasticita

Jednoduché plastické deformace můžeme docílit např. přidáním tohoto kódu za řádek 16:

```
1 if (del.length()<0.8*btSqrt(l.m_c1))
2 {
3 l.m_c1 = 0.8*0.8*l.m_c1;
4 }
```

Kód říká, že jestliže je vzdálenostní vazba „stlačená“ o víc jak 20% oproti klidové vzdálenosti, tak klidovou vzdálenost např. o těch 20% zkrátíme. Po přidání tohoto kódu skutečně zůstávají „plasticky“ zdeformované. Vyzkoušel jsem vytvořit předek auta pomocí Bullet soft body. Definici sítě jsem naprogramoval v Matlabu a definici sítě exportoval do souboru, který načítal Bullet. Předek auta je soft body objekt tvořen vrcholy a spojnicemi. Zbytek auta je tvořen pevnými tělesy. Bullet nabízí spojení obou modulů, takže i spojení pevných těles s deformovatelnými tělesy. Na obrázku 2.1 je předek před deformací a po deformaci a skutečně zůstává plasticky zdeformovaný.



(a) Před nárazem

(b) Po nárazu

Obrázek 2.1: Jednoduchá plasticita v Bulletu

2.3 Velké deformace popsané pomocí mechaniky kontinua

Při popisu deformací pevných těles pomocí mechaniky kontinua se nejčastěji používá tzv. Lagrangeův popis, t.z. vychází se z referenční (klidové) konfigurace tělesa a hodnoty veličin se vztahují k materiálovým bodům tělesa v referenční konfiguraci. Těleso před deformací zaujímá množinu $\Omega_0 \in \mathbb{R}^3$. Představme si, že chceme např. modelovat prohnutí elastického kvádru vlastní vahou, který je na jedné straně připevněný ke stěně. Ω_0 pak bude nezdeformovaný kvádr, např. množina $[0,5] \times [0,1] \times [0,1]$ (souřadnice (x,y,z)). Časově závislou deformaci tělesa popisuje zobrazení $\chi : \Omega_0 \times [0,T] \longrightarrow \mathbb{R}^3$, t.j. v čase $t \in [0,T]$ zaujímá materiálový bod X z původní konfigurace Ω_0 ($X \in \Omega_0$) polohu $\chi(X,t)$. Často se zavádí a pracuje s vektorem posunutí, který vede z materiálového bodu do jeho aktuální

pozice $u(X,t) = \chi(X,t) - X$. Mechanika kontinua se pak snaží zodpovědět otázku, jak vypadá vektor posunutí elastického tělesa, jestliže známe

- **Externí síly působící na těleso**
  Tyto síly působí na všechny body tělesa a zadávají se pomocí vektorové hustoty sil $b(X,t)$. V našem příkladu s kvádrem se jedná pouze o gravitaci, $b(X,t) = \rho_0(X,t)\vec{g}$, kde ρ_0 je hustota tělesa v počáteční konfiguraci a $\vec{g} = (0,0,-9.81)$.

- **Povrchové síly**
  Tyto síly působí na tzv. Neumannově hranici tělesa $\partial\Omega_N \subseteq \partial\Omega_0$ a jsou vztaženy na jednotkovou plošku hranice. Zadávají se opět vektorovou hustotou $T(X,t)$. V případě deformace vlastní vahou takové síly neuvažujeme.

- **Hraniční podmínky pro posunutí neboli Dirichletovy okrajové podmínky**
  Jedná se o předepsaný vektor posunutí na Dirichletově hranici $\partial\Omega_D \subseteq \partial\Omega_0$. V našem příkladě by podmínka na připevněnou stranu kvádru vypadala $u = 0$ na $\partial\Omega_D = [0] \times [0,1] \times [0,1]$, což je levá strana kvádru ve tvaru čtverce.

Z fyzikálního zákona zachování hybnosti se dá odvodit parciální diferenciální rovnice pro vektor posunutí u popisující pohyb elastického kontinua, která vypadá následovně

$$\rho_0 \frac{\partial^2 u}{\partial t^2} = Div(P) + \rho_0 b, \tag{2.1}$$

kde P je tzv. první Piola Kirchhoffův tenzor napětí. Rovnice je doplněná o počáteční a okrajové podmínky.

$$u(X,0) = u_0(X), \ \frac{\partial u(X,0)}{\partial t} = v_0(X) \text{ v } \Omega_0 \tag{2.2}$$

$$u(X,t) = g(X,t) \text{ na } \Omega_D, \ P(X,t)N(X) = T(X,t) \text{ na } \Omega_N, \tag{2.3}$$

kde $N(X)$ je jednotková vnější normála v bodě X.

2.3.1 Nelinearity, konstitutivní vztahy

K tomu, abychom dostali parciální diferenciální rovnici v hledaných posunutích, musíme zakomponovat materiálové vztahy a vztahy mezi mírou deformace a posunutím. Tzv. druhý Piola-Kirchhoffův tenzor napětí S je s prvním Piola-Kirchhoffovým tenzorem napětí svázán vztahem

$$P = FS, \tag{2.4}$$

kde F je deformační gradient

$$F = \frac{\partial \chi}{\partial X} = \nabla\chi, \text{což napsáno v indexech znamená } \ F_{ij} = \frac{\partial \chi_i}{\partial X_j}, \quad i,j = 1,\ldots,3.$$

Platí, že

$$F = I + \nabla u.$$

Tenzory napětí a tenzory deformace jsou svázány tzv. konstitutivními vztahy. Co se týče velkých deformací řešených v rámci počítačové grafiky, tak většinou se uvažuje pouze jeden z nejjednodušších konstitutivních vztahů a sice

$$S = \lambda tr(E)I + 2\mu E. \tag{2.5}$$

Tímto konstitutivním vztahem mezi druhým Piola-Kirchhoffovým tenzorem a tzv. Greenovým tenzorem velkých deformací E je modelován hyperelastický materiál, který se nazývá St.Venant-Kirchhoffův materiál. Greenův tenzor velkých deformací je používaná „míra deformace“ u velkých deformací a pomocí posunutí je vyjádřen vztahem

$$E = \frac{1}{2}\left(\nabla u + (\nabla u)^T + (\nabla u)(\nabla u)^T\right)$$

Dá se napsat např. i jako

$$E = \frac{1}{2}\left(C - I\right),$$

kde $C = F^T F$ je tzv. pravý Cauchyho-Greenův tenzor. Greenův tenzor velkých deformací modeluje geometrickou nelinearitu vznikající při velkých deformacích. Je invariantní vůči posunutím a rotacím tělesa, na rozdíl od tenzoru malých deformací

$$\varepsilon = \frac{1}{2}\left(\nabla u + (\nabla u)^T\right),$$

který se používá v tzv. lineární elasticitě a je linearizací Greenova tenzoru pro malé deformace. V lineární elasticitě se pro izotropní materiál používá lineární vztah mezi tenzorem malých deformací a Cauchyho tenzorem napětí σ, který má tvar

$$\sigma = \lambda tr(\varepsilon)I + 2\mu\varepsilon,$$

kde λ a μ jsou tzv. Lamého koeficienty. Používají se také tzv. Youngův modul pružnosti E a Poissonova konstanta ν. Vztahy mezi nimi a Lamého koeficienty jsou následující.

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

Vztah pro St.Venant-Kirchhoffův materiál je přímočaré převedení tohoto lineárního materiálového vztahu na Greenův tenzor a velké deformace. Nelineární vztah mezi tenzorem velkých deformací a ∇u vnáší do rovnic tzv. geometrickou nelinearitu. Další, tzv. materiálová nelinearita by vznikla, kdybychom uvažovali

nelineární konstitutivní vztah mezi tenzorem napětí a tenzorem deformace. Třetí druh nelinearit vzniká při formulaci kontaktních podmínek. Geometrická nelinearita způsobí, že dostaneme nelineární parciální diferenciální rovnici popisující pohyb elastického tělesa. Po diskretizaci pomocí metody konečných prvků dostaneme nelineární obyčejné diferenciální rovnice s obecně velkým počtem neznámých. Proto je metoda konečných prvků obtížně použitelná v grafických real time aplikacích. V poslední době se ale objevily způsoby a snahy, jak metodu konečných prvků využít i na tyto aplikace. Jednou z nich je tzv. korotační metoda konečných prvků. Jestliže uvažujeme tenzor malých deformací

$$\varepsilon = \frac{1}{2}\left(\nabla u + (\nabla u)^T\right) = \frac{1}{2}\left(F + F^T\right) - I,$$

pak není invariantní vůči rotacím. Uvažujeme-li pouze rotaci bez deformací $F = R$ (kde R je ortogonální matice), pak dosazením do vztahu nahoře vidíme, že nám vyjde nenulová deformace a s tím i nenulový tenzor napětí. V tělese tak vznikají nefyzikální síly způsobující např. nepřirozené zvětšování tělesa. Korotační metoda se snaží využít jednoduchý tvar tenzoru malých deformací a vyextrahovat z deformace rotaci.

2.4 Korotační lineární metoda konečných prvků

Implementace korotační lineární (nebo jen lineární bez extrakce rotace) MKP se dá nalézt např. v projektu Open cloth `code.google.com/p/opencloth`. Aby se dala metoda konečných prvků počítat co nejrychleji, používají se v počítačové grafice k diskretizaci v podstatě jen lineární funkce na tetrahedronové síťi. Čerpám ze článku [20], který popisuje návrh kompletní implementace korotační metody. Pro diskretizaci pomocí lineárních funkcí na tetrahedronové síti je situace následující. Nechť referenční souřadnice vrcholů tetrahedronu jsou X_1, X_2, X_3, X_4. Aktuální pozice nechť jsou x_1, x_2, x_3, x_4. Deformační gradient je pak roven

$$\mathbf{F} = \frac{\partial x}{\partial X} = [x_2 - x_1, x_3 - x_1, x_4 - x_1][X_2 - X_1, X_3 - X_1, X_4 - X_1]^{-1}$$

Chceme používat tenzor malých deformací, který ale není invariantní vůči rotacím a proto se nedají dobře modelovat velké deformace, u kterých dochází k velkým rotacím. Tenzor malých deformací je

$$\epsilon = \frac{1}{2}(\mathbf{F} + \mathbf{F}^T) - \mathbf{I}$$

Dekompozici deformačního tenzoru na rotaci a čistou deformaci získáme např. pomocí polárního rozkladu:

$$\mathbf{F} = \mathbf{Q}\mathbf{A},$$

kde $\mathbf{Q}$ je ortonormální a $\mathbf{A}$ je symetrická matice. Je známo, že takový rozklad vždy existuje. Zavedeme korotační deformační gradient

$$\tilde{\mathbf{F}} = \mathbf{Q}^T\mathbf{F}$$

a korotační tenzor deformace

$$\tilde{\epsilon} = \frac{1}{2}(\tilde{\mathbf{F}} + \tilde{\mathbf{F}}^T) - \mathbf{I}$$

Lineární izotropický konstitutivní model pro Cauchyho tenzor napětí použijeme pro korotační tenzor deformace

$$\sigma = \lambda tr(\tilde{\epsilon})\mathbf{I} + 2\mu\tilde{\epsilon}$$

Dostaneme lineární soustavu obyčejných diferenciálních rovnic

$$\mathbf{Ma} + \mathbf{Cv} + \mathbf{K}(\mathbf{x} - \mathbf{X}) = \mathbf{f} \tag{2.6}$$

kde $\mathbf{x}$ je vektor souřadnic vrcholů v aktuální konfiguraci, $\mathbf{X}$ vektor vrcholů v referenční konfiguraci a $\mathbf{v}$ a $\mathbf{a}$ jsou vektory rychlostí a zrychlení. Vektor $\mathbf{f}$ reprezentuje vnější síly působící na těleso. Matice $\mathbf{M}$, $\mathbf{C}$ a $\mathbf{K}$ jsou matice hmotnosti, tlumení a tuhosti.

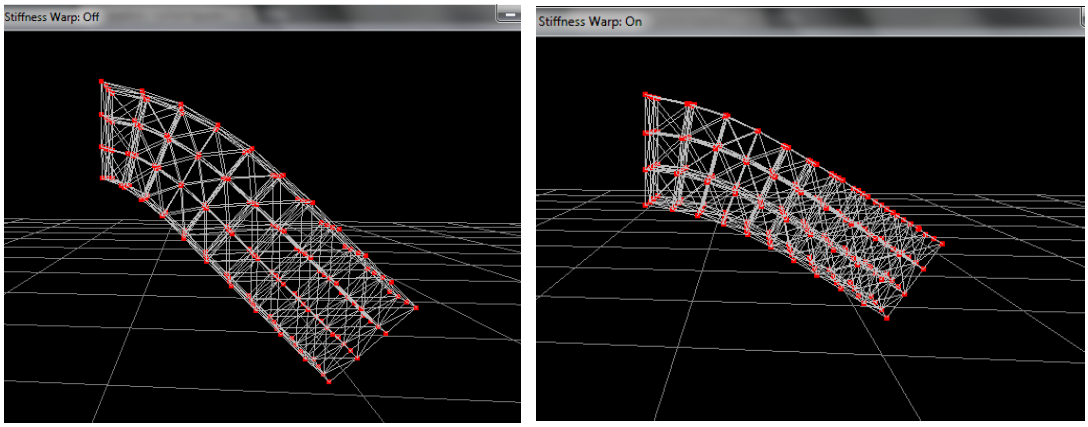
Autoři článnku [20] metodu implementovali konkrétně v enginu Dynamic molecular matter firmy Pixelux entertainment. V článku navrhují následující numerickou diskretizaci a řešení obdržené soustavy rovnic. Pro nové rychlosti a pozice máme

$$\mathbf{v}^+ = \mathbf{v} + \Delta t\mathbf{a}$$
$$\mathbf{x}^+ = \mathbf{x} + \Delta t\mathbf{v}$$

Dosazením do rovnice 2.6 dostaneme lineární rovnici pro nové rychlosti $\mathbf{v}^+$. Následující soustavu rovnic navrhují autoři řešit např. pomocí metody konjugovaných gradientů

$$\left(\mathbf{M} + \Delta t\mathbf{C} + (\Delta t)^2\mathbf{K}\right)\mathbf{v}^+ = \Delta t\mathbf{f} + \mathbf{Mv} - \Delta t\mathbf{K}(\mathbf{x} - \mathbf{X})$$

Oproti modelu hmotný bod - pružina máme u korotační metody možnost vygenerovat síť, nastavit materiálové vlastnosti a metoda už vše řeší za nás. Naproti tomu u hmotných bodů s pružinami chování objektu závisí podstatně na sestavení pružinkové sítě. Jak jsme zmínili, dá se nalézt implementace této metody např. v projektu OpenCloth `code.google.com/p/opencloth`. Warpování tuhosti se dá v kódu vypnout. Na obrázku 2.2 je nalevo těleso s vypnutou korotační metodou a napravo se zapnutou. To znamená, že se nalevo používá tenzor malých deformací, který u větší deformace (s výraznější rotací) zapříčiní nepřirozené zvětšení objemu tělesa.



(a) Tenzor malých deformací

(b) Korotační tenzor deformace

Obrázek 2.2: Porovnání vypnuté a zapnuté korotace

Kapitola 3

Redukce modelu

Vraťme se nyní k rovnicím, které popisují plnou elasticitu, tedy obahující Green-Lagrangeův tenzor deformace.

3.1 Redukce rovnic dynamického systému

Dynamickou rovnici elastického tělesa, které podléhá velkým deformacím, je potřeba řešit numericky. Nejčastěji se používá metoda konečných prvků nebo metoda konečných diferencí. U metody konečných prvků se provede prostorová diskretizace tělesa a dynamická parciální diferenciální rovnice přejde na soustavu obyčejných diferenciálních rovnic ve tvaru

$$M\ddot{u} + D(u,\dot{u}) + R(u) = f,$$

kde M je tzv. matice hmotnosti, D matice tlumení, $R(u)$ je vektor vnitřních sil a f vektor vnějších sil působících na těleso. Neznámá vektorová funkce $u : [0,T) \to \mathbb{R}^n$ většinou reprezentuje posunutí vrcholů sítě, kterou je těleso reprezentováno, n představuje počet těchto vrcholů neboli obecně počet stupňů volnosti systému. V praxi se používají husté sítě a počet stupňů volnosti n tak bývá velké číslo, a proto dostaneme velkou soustavu obyčejných diferenciálních rovnic, kterou je potřeba řešit.

Dimenzionální redukce modelu je technika jak zrychlit simulace dynamických systémů, které jsou popsány obyčejnými diferenciálními rovnicemi. Řešení rovnic aproximujeme řešeními z nějakého podprostoru, který má mnohem menší dimenzi než systém původní. Jestliže je $u : [0,T) \to \mathbb{R}^n$, pak zavedeme redukované souřadnice $q : [0,T) \to \mathbb{R}^r$, $r << n$ transformací $u = u_0 + Uq$, kde ve sloupcích matice U jsou vektory báze aproximačního podprostoru a u_0 je nějaké počáteční posunutí, ve kterém jsme se rozhodli soustavu rovnic redukovat. Pokud uvažujeme časově neměnnou bázi v matici U, přejde původní systém rovnic

$$M\ddot{u} + D(u,\dot{u}) + R(u) = f$$

na menší systém rovnic

$$\overline{M}\ddot{q} + \overline{D}(q,\dot{q}) + \overline{R}(q) = \overline{f},$$

kde

$$
\begin{aligned}
\overline{M} &= U^T M U \\
\overline{D}(q,\dot{q}) &= U^T D(u_0 + Uq, U\dot{q}) \\
\overline{R}(q) &= U^T R(u_0 + Uq) \\
\overline{f} &= U^T f
\end{aligned}
$$

3.2 Volba redukované báze

Hlavní otázkou u redukce modelu tedy zůstává volba redukované báze. Zřejmě je potřeba mít nějakou množinu posunutí (v případě, že řešíme deformace), ze které bychom vhodnou redukovanou bázi vybrali. Taková báze by měla nejlépe reprezentovat „typické“ deformace tělesa. Vybírat bychom mohli ze spočtených řešení, která jsme obdrželi plnou simulací pomocí nějaké metody a např. za různých okrajových/počátečních podmínek nebo i z nějakých experimentálně zjištěných dat. Takový přístup by se dal nazvat aposteriorní.

Lepší by ale bylo, kdybychom předem nemuseli nic počítat a mohli bychom redukovanou bázi vypočítat rovnou na základě nějakých charakteristik systému. Na tento apriorní způsob se podívejme nejdříve.

3.2.1 Apriorní redukovaná báze

Tento přístup je popsán ve článku [9], kterého se budeme držet, ale metodu používá a popisuje např. i Jernej Barbič, autor softwaru Vega FEM [4], který dokáže urychlit simulace velkých deformací objektů počítaných pomocí metody konečných prvků právě na základě redukce a výběru redukované báze apriorně. Tato posunutí se počítají z tzv. modální analýzy systému a pro zpřesnění se přidají ještě tzv. modální derivace.

Modální analýza

Stejně jako např. struna má vlastní kmity s vlastními frekvencemi, tak i deformovatelné těleso má jisté specifické deformační módy, do kterých se deformuje „nejpřirozeněji“ a s co nejmenší potřebnou energií. Mějme matici tuhosti systému K v referenční konfiguraci a matici hmotnosti M. Tzv. lineární modální analýza hledá vlastní frekvence a vlastní tečné deformační módy. Eliminujeme Dirichletovy okrajové podmínky (odpovídající fixovaným vrcholům konečněprvkové sítě) a obdržíme matice $\overline{K}$ a $\overline{M}$. Pro jednoduchost budeme dále čárky nad maticemi vynechávat. Řešíme zobecněný problém vlastních čísel

$$K\Phi = \lambda M\Phi$$

Hledáme k nejmenších vlastních čísel a vlastních vektorů. Vlastní čísla jsou vlastní (přirozené) frekvence ω_i na druhou.

$$\lambda_i = \omega_i^2$$

Tyto vlastní vektory se většinou normují pomocí matice hmotnosti, jako i v článku [9]. Soustava se dá napsat i následovně

$$\left(K - \omega_p^2 M\right) \Phi_p = 0, \quad p = 1, \ldots, N.$$

Hledáme nejmenší frekvence, jelikož k jejich vybuzení budou potřeba menší energie. Redukovanou bázi tedy můžeme zvolit z módů, které mají P nejmenších frekvencí ω_p a dostaneme lineární aproximaci posunutí v tzv. modálních souřadnicích α_p (p = 1,...,P), což jsou parametry v rozvoji do tečných (lineárních) módů.

$$u = \sum_{p=1}^{P} (\Phi_p \alpha_p) = Uq$$

kde $q = (\alpha_1, ..., \alpha_P)^T$. Matice U obsahuje ve sloupcích vlastní vektory Φ_p. V tomto případě máme $u_0 = 0$.

Modální derivace

Pokud vezmeme za redukovanou bázi tečné módy z modální analýzy a za redukované souřadnice vezmeme modální souřadnice, bude takto redukovaný systém popisovat dobře pouze malé deformace okolo referenční konfigurace, což dokládají i výše zmíněné dva články. Redukovaná báze se dá vylepšit o tzv. modální derivace. Jedná se o analogii Taylorova rozvoje posunutí, kde tečné lineární módy by představovaly aproximace (derivace) prvního řádu a modální derivace by pak představovaly aproximace (derivace) druhého řádu. V následujícím odstavci popíšeme, jak se dají modální derivace odvodit.

Tečné módy závisejí na tečné matici tuhosti pro nějaké posunutí (u modální analýzy jsem mluvil o referenční kofiguraci, t.j. $u = 0$, ale stejný výpočet můžeme provést pro libovolné posunutí a matici tuhosti pro dané posunutí). Tečné módy proto můžeme chápat jako funkce posunutí.

$$u = \sum_{p=1}^{N} (\Phi_p(u) \alpha_p) = U(u)q \tag{3.1}$$

Aproximované posunutí u je funkcemi modálních souřadnic α_p a tedy můžeme chápat i tečné módy jako funkce modálních souřadnic $\Phi_i = \Phi_i(\alpha_p)$.

Předpokládejme, že můžeme posunutí rozvinout do Taylorova polynomu 2. řádu v modálních souřadnicích okolo referenční konfigurace (t.j. $q = 0$, ve smyslu $\alpha_i = 0$, $i = 1, \ldots, N$). Obecně ale můžeme rozvíjet kolem nějaké konfigurace u_0.

$$u = \sum_{i=1}^{N} \frac{\partial u}{\partial \alpha_i}(q = 0)\alpha_i + \frac{1}{2} \sum_{p=1}^{N} \sum_{r=1}^{N} \frac{\partial^2 u}{\partial \alpha_p \partial \alpha_r}(q = 0)\alpha_r \alpha_p$$

Z (3.1) zderivováním dostaneme

$$\frac{\partial u}{\partial \alpha_i} = \Phi_i + \sum_{r=1}^{N} \left(\frac{\partial \Phi_r}{\partial \alpha_i} \alpha_r \right)$$

$$\frac{\partial^2 u}{\partial \alpha_p \partial \alpha_r} = \frac{\partial \Phi_r}{\partial \alpha_p} + \frac{\partial \Phi_p}{\partial \alpha_r} + \sum_{t=1}^{N} \left(\frac{\partial^2 \Phi_t}{\partial \alpha_p \partial \alpha_r} \alpha_t \right)$$

$$\frac{\partial u}{\partial \alpha_i}(q = 0) = \Phi_i$$

$$\frac{\partial^2 u}{\partial \alpha_p \partial \alpha_r}(q = 0) = \frac{\partial \Phi_r}{\partial \alpha_p} + \frac{\partial \Phi_p}{\partial \alpha_r}$$

Výrazy $\frac{\partial \Phi_r}{\partial \alpha_p}$ nazýváme tzv. modální derivace. Vidíme také, že první derivace (aproximace v Taylorově rozvoji) odpovídají právě tečným módům z modální analýzy.

Jak vypočítat numericky modální derivace

V článku [9] autor popisuje 3 metody, pomocí nichž se dají modální derivace vypočítat numericky.

1. možnost

Jesliže se rozhodneme modální derivaci počítat pomocí konečné diference

$$\frac{\partial \Phi_r}{\partial \alpha_p} \approx \frac{\Phi_r(u_0 + \Phi_p \delta\alpha_p) - \Phi_r(u_0)}{\delta\alpha_p},$$

je potřeba řešit dva problémy vlastních čísel. $\Phi_r(u_0)$ a $\Phi_r(u_0 + \Phi_p \delta\alpha_p)$ představují vlastní vektory příslušné k tečným maticím tuhosti $K(u_0)$ a $K(u_0 + \Phi_p \delta\alpha_p)$. $\delta\alpha_p$ je potřeba brát dostatečně malé, aby dobře aproximovalo derivaci, ale ne zas moc malé, což by dělalo problém v konečné počítačové aritmetice.

2. možnost

Zderivujeme-li soustavu na výpočet tečných módů

$$\left(K - \omega_r^2 M\right) \Phi_r = 0, \quad K = K(\alpha_p), \Phi_r = \Phi_r(\alpha_p)$$

podle modální souřadnice α_p, dostaneme soustavu

$$\left(K - \omega_r^2 M\right) \frac{\partial \Phi_r}{\partial \alpha_p} = \left(\frac{\partial \omega_r^2}{\partial \alpha_p} M - \frac{\partial K}{\partial \alpha_p} \right) \Phi_r, \tag{3.2}$$

kterou je potřeba řešit. Výrazem $\frac{\partial K}{\partial \alpha_p}$ rozumíme derivaci matice tuhosti vzhledem k p-té modální souřadnici.

$$\frac{\partial K}{\partial \alpha_p} \approx \frac{K(u_0 + \Phi_p \delta\alpha_p) - K(u_0)}{\delta\alpha_p}$$

Řešení soustavy (3.2) Při modální analýze a počítání tečných módů se může stát, že k jedné vlastní frekvenci ω_r náleží s tečných módů. V takovém případě má matice $K - \omega_r^2 M$ rank $n - s$ a její jádro tvoří tyto tečné módy. Jak je známo z lineární algebry, obecné řešení soustavy (3.2) tvoří množina všech řešení homogenní a jedno vybrané řešení nehomogenní soustavy.

$$\frac{\partial \Phi_r}{\partial \alpha_p} = \frac{\partial \Phi_r}{\partial \alpha_p}_{hom.} + \frac{\partial \Phi_r}{\partial \alpha_p}_{nehom.}$$

Bez újmy na obecnosti můžeme předpokládat, že $s = 2$ a že tedy k jedné frekvenci ω_r náleží 2 tečné módy Φ_r a Φ_{r+1}. Uvažujme transformaci T takovou, že

$$\frac{\partial \Phi_r}{\partial \alpha_p}_{nehom.} = T \widetilde{\frac{\partial \Phi_r}{\partial \alpha_p}}$$

kde T je jednotková matice mající v k-tém sloupci vlastní vektor Φ_r a v l-tém sloupci $(k < l)$ vlastní vektor Φ_{r+1}. Pomocí transformace T transformujeme soustavu na

$$T^T \left(K - \omega_r^2 M\right) T \widetilde{\frac{\partial \Phi_r}{\partial \alpha_p}} = T^T \left(\frac{\partial \omega_r^2}{\partial \alpha_p} M - \frac{\partial K}{\partial \alpha_p}\right) \Phi_r \tag{3.3}$$

kde k-tá a l-tá rovnice jsou

$$\begin{aligned} 0 &= \Phi_r^T M \Phi_r \frac{\partial \omega_r^2}{\partial \alpha_p} - \Phi_r^T \frac{\partial K}{\partial \alpha_p} \Phi_r \\ 0 &= \Phi_{r+1}^T M \Phi_r \frac{\partial \omega_r^2}{\partial \alpha_p} - \Phi_{r+1}^T \frac{\partial K}{\partial \alpha_p} \Phi_r \end{aligned}$$

Vlastní vektory jsou normalizované vzhledem k matici hmotnosti a proto se rovnice zjednodušší na

$$\begin{aligned} \frac{\partial \omega_r^2}{\partial \alpha_p} &= \Phi_r^T \frac{\partial K}{\partial \alpha_p} \Phi_r \\ 0 &= \Phi_{r+1}^T \frac{\partial K}{\partial \alpha_p} \Phi_r \end{aligned}$$

Druhá rovnice vyjadřuje podmínku, kterou musí vlastní vektory splňovat. Z první rovnice dosadíme do (3.3) a dostaneme

$$\left(K - \omega_r^2 M\right)_* \left\{ \widetilde{\frac{\partial \Phi_r}{\partial \alpha_p}} \right\}_* = \left\{ \left(M \Phi_r \Phi_r^T - I\right) \frac{\partial K}{\partial \alpha_p} \Phi_r \right\}_*$$

kde $(\ldots)_*$ značí matici, ze které jsme vypustili k-tý a l-tý řádek a $\{\ldots\}_*$ značí vektor, ve kterém jsme vypustili k-tý a l-tý element. Jestliže tuto lineární soustavu vyřešíme a výsledek $\left\{ \widetilde{\frac{\partial \Phi_r}{\partial \alpha_p}} \right\}_*$ doplníme o k-tý a l-tý element rovný nule, získáme hledané partikulární řešení nehomogenní soustavy $\frac{\partial \Phi_r}{\partial \alpha_p}_{nehom.}$.

3. možnost

Poslední a nejjednodušší možností je počítání modálních derivací zanedbáním inerciálních členů v soustavě (3.2), t.j zanedbání matice hmotnosti. V článku [9] autor tuto metodu zdůvodňuje tím, že přenásobení matice hmotnosti libovolným nenulovým faktorem γ nezmění tvar modálních derivací.

$$\frac{\partial \Phi_r}{\partial \alpha_p} = -K^{-1}\frac{\partial K}{\partial \alpha_p}\Phi_r$$

Tento způsob je zjevně i z numerického hlediska nejméně náročný, protože dostaneme přímo lineární rovnici, a využívá ho i zmíněný software Vega FEM [4]. Dále je ukázáno, že takto vypočítané modální derivace vykazují následující symetrie:

$$\frac{\partial \Phi_r}{\partial \alpha_p} = \frac{\partial \Phi_p}{\partial \alpha_r},$$

což je výhodné při počítání rozvoje

$$\frac{\partial^2 u}{\partial \alpha_p \partial \alpha_r}(\alpha = 0) = \frac{\Phi_r}{\alpha_p} + \frac{\Phi_p}{\alpha_r} = 2\frac{\Phi_r}{\alpha_p} = 2\frac{\Phi_p}{\alpha_r},$$

kde nemusíme počítat obě derivace, ale stačí počítat jen jednu z nich.

3.2.2 Aposteriorní redukovaná báze

Jinou možností je vybírat redukovanou bázi z předem (offline) vypočítaných deformací (posunutí). K tomu se používá metoda zvaná POD (zkratka z proper orthogonal decomposition). Dá se nalézt i pod názvy principal component analysis (PCA), my budeme používat zkratku POD. Tuto metodu využívá i software Vega FEM při vybírání modálních derivací. Jejich počet totiž s počtem tečných módů roste kvadraticky a proto se ještě používá POD na vybírání modálních derivací, které charakteristické deformace popisují „nejlépe“.

Proper orthogonal decomposition (POD)

Tato metoda se obecně používá k aproximaci dat s velkou dimenzí vybranými daty, které mají dimenzi menší.

Motivace

Následující motivace je převzatá z článku [8]. Mějme funkci $z(x,t)$, kterou chceme aproximovat na nějaké množině Ω konečnou sumou v separovaných proměnných

$$z(x,t) = \sum_{k=1}^{M} a_k(t)\phi_k(x).$$

Chtěli bychom, aby taková suma v limitě konvergovala k funkci z v nějakém smyslu a libovolně přesně ji aproximovala. Máme na výběr z více možností, např. Fourierův rozvoj nebo Legendrovy polynomy atd. K nějaké takové bázi vybraných

funkcí ϕ_k obdržíme obecně různé posloupnosti a_k. Volba určité báze se týká i metody POD. U POD požadujeme, aby byly funkce orthogonální, t.j.

$$\int_\Omega \phi_{k_1}(x)\phi_{k_2}(x)dx = \begin{cases} 1, & k_1 = k_2 \\ 0, & k_1 \neq k_2 \end{cases}$$

a aby pro zvolené M byla chyba aproximace funkce z minimální ve smyslu nejmenších čtverců.

Konečně - dimenzionální případ

Mějme matici $X \in \mathbb{R}^{m\times n}$. Matice může reprezentovat např. naměřená m-dimenzionální data v n exemplářích, např. n digitálních obrázků, z nichž každý je reprezentovaný m-dimenzionálním vektorem. V takovém případě obvykle analyzujeme velké množství obrázků a $m << n$. Nebo může matice reprezentovat výstup z počítání deformací metodou konečných prvků, kde m je počet stupňů volnosti a n značí výpočet v n různých časech nebo pro n různých okrajových podmínek. Počet stupňů volnosti v metodě konečných prvků bývá velký a počet výpočtů oproti tomu by byl znatelně menší $m >> n$. S tím souvisí metoda snapshotů, viz pozdější text. Metoda POD předpokládá, že vzorky/sloupce v matici X mají nulový součet, je tedy potřeba od nich odečíst jejich průměr. Mějme transformaci do orthogonálních souřadnic prostoru $\mathbb{R}^m$ pomocí ortogonální matice $P \in \mathbb{R}^{m\times m}$

$$Y = PX.$$

Tato rovnice se dá chápat i jako konečně dimenzionální analogie sumace funkcí výše.

POD se dá interpretovat jako hledání takové orthogonální transformace, aby byla matice YY^T diagonální, jako maximalizace variance projektovaných dat nebo právě aby se matice Y lišila od matice X s nejmenší chybou ve smyslu nejmenších čtverců (viz Low rank aproximation dole). Řešení spočívá v nalezení vlastních čísel matice $XX^T \in \mathbb{R}^{m\times m}$. Tato matice je symetrická a pozitivně definitní. Jelikož jsme od dat X odečetli jejich střední hodnotu, odpovídá tato matice tzv. matici kovariance. Pro hledání vlastních vektorů se používá např. singulární rozklad.

Metoda snapshotů Matice XX^T je $m \times m$ a v případě, že $m >> n$ to může být nepříjemný problém, neboť nejsme schopni takovou matici třeba ani uchovat v paměti. Proto se použije následující trik. Vytvoříme menší matici $X^TX \in \mathbb{R}^{n\times n}$. Pro její vlastní vektory v platí

$$\begin{aligned} (X^TX)v &= \lambda v \\ X(X^TX)v &= X\lambda v \\ (XX^T)Xv &= \lambda Xv \end{aligned}$$

To znamená, že vlastní vektory matice XX^T pak dostaneme transformací $\overline{v} = Xv$.

Hledání POD módů se dá nalézt i pod názvem Low rank aproximation a dá se řešit pomocí singulárního rozkladu.

Low rank aproximation Nechť $Y \in \mathbb{R}^{m\times n}$, $rank(Y) = r$. Matice může představovat např. n-krát naměřené m-dimenzionální hodnoty. Při tzv. low rank aproximaci se snažíme tuto matici aproximovat maticí s menším rankem. Matematicky můžeme problém formulovat jako

$$\min_{\hat{Y}\in\mathbb{R}^{m\times n}} \left\{ \|Y - \hat{Y}\|_F,\ rank(\hat{Y}) = k,\ 1 \le k < r \right\}.$$

Omezení na matice je tedy zvolený rank k. Maticová Frobeniova norma je definována následovně

$$\|Y\|_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} Y_{ij}^2}.$$

Problém low rank aproximace matice má explicitní řešení pomocí tzv. singulárního rozkladu. Nechť

$$Y = U\Sigma V^T$$

je singulární rozklad matice Y. Σ je diagonální matice a na diagonále má singulární čísla : $\sigma_1 \ge \sigma_2 \ge ... \ge \sigma_r \ge 0$. Optimální k rank aproximaci získáme ponecháním k největších singulárních čísel v matici Σ a vynulováním $r - k$ singulárních čísel: $\sigma_{k+1} = ... = \sigma_r = 0$. Takto modifikovaná matice Σ_k pak dá optimální k rank aproximaci jako

$$Y_k = U\Sigma_k V^T.$$

Minimální chyba je dána

$$\|Y - Y_k\|_F = \sqrt{\sigma_{k+1}^2 + ... + \sigma_r^2}.$$

Tento výsledek je znám pod názvem Eckart-Young-Mirsky theorem.

Singulární rozklad

Existence singulárního rozkladu. Nechť $Y \in \mathbb{R}^{m\times n}$. Pak existují unitární matice $U \in \mathbb{R}^{m\times m}$ a $V \in \mathbb{R}^{n\times n}$ takové, že

$$Y = U\Sigma V^T,$$

kde

$$\Sigma = \begin{pmatrix} S & 0_{k\times(n-k)} \\ 0_{(m-k)\times k} & 0_{(m-k)\times(n-k)} \end{pmatrix} \in \mathbb{R}^{m\times n}$$

$k \le min(m,n)$, $S = diag(\sigma_1,...\sigma_k)$ a platí $\sigma_1 \ge \sigma_2 \ge ... \ge \sigma_k \ge 0$.

Článek [11] popisuje, jak redukovanou bázi nalézt aposteriorně. K aposteriorní redukci je používána metoda POD. Autoři představují ještě další lehké úpravy této metody, které celý proces mohou ještě zlepšit a urychlit. Stěžejní je ale vždy metoda POD.

A posteriori redukce

Během preprocesní části by se měly nejlépe vypočítat všechny možné interakce (působení sil) s deformovatelným tělesem. Toto samozřejmě není možné, takže se musí vybrat jen některé, ale může jich být hodně. Nebo se mohou analyzovat data získaná např. experimentálně, nebo uložíme výsledky v různých časových krocích nějakého dynamického problému. Tuto možnost následně vyzkoušíme. Jelikož je těchto dat velké množství, musí se nějak redukovat a vybrat z nich „charakteristické“ deformace. I když redukci děláme např. za účelem interakce s objektem v reálném čase (dynamický problém), během preprocesní fáze můžeme řešit zatížení jako statické a dostaneme soustavu úloh pro zatížení $F_S, S \in N$, které jsou tvaru

$$K(u_S)u_S = F_S.$$

Získaná posunutí u_S pak uspořádáme do matice $X \in \mathbb{R}^{N \times S}$. Pomocí PCA můžeme pak takovou matici s daty redukovat. Hledáme redukovanou bázi, která generuje podprostor $H \subset \mathbb{R}^N$, $dim(H) = m$. Hledanými vektory bude m vlastních vektorů, náležících k m největším vlastním číslům kovariantní matice $C_d = \bar{X}\bar{X}^T$, kde

$$\bar{X} = [\bar{u}_1, \ldots, \bar{u}_S]$$

a

$$\bar{u}_s = u_s - \frac{1}{S}\sum_{i=1}^{S} u_i.$$

Z báze vytvoříme matici U a výpočty provádíme v redukovaných souřadnicích q_s. Aproximovaná posunutí dostaneme ze vztahu

$$\tilde{u}_s = Uq_s + \bar{u},$$

kde

$$\bar{u} = \frac{1}{S}\sum_{i=1}^{S} u_i.$$

Pro výpočty dynamiky v redukovaných souřadnicích je potřeba v paměti uchovávat matice $U \in \mathbb{R}^{N \times m}, Q \in \mathbb{R}^{m \times S}$, jejíž sloupce jsou q_s a vektor $\bar{u} \in \mathbb{R}^N$. Počet uchovávaných dat je $m(N + S) + N$ oproti původním NS.

3.3 Numerické výpočty

K vyzkoušení redukce použijeme postupy popsané v [9], kde se dají nalézt i skripty implementované v Matlabu, které jsem modifikoval ke svým potřebám. V tomto článku se autor zabývá možnostmi, jak redukovat rovnice nelineárního dynamického systému, které vzniknou diskretizací pomocí metody konečných prvků. Redukovaná báze je zde sestavená buď pouze z tečných módů nebo z tečných módů a modálních derivací. Autor představuje možnost jak aktualizovat redukovanou bázi v určitém čase a posunutí na základě chyby mezi redukovaným a neredukovaným systémem. Dále je zkoumána linearizace nelineárního systému a její integrace a redukce zlinearizovaného systému. Celý problém je demonstrován na příkladu 2D tyčové soustavy, která je přichycená na levém kraji a je vystavena působení předepsané síly. Tyčová soustava má 8 vrcholů, které jsou spojeny 13 tyčovými elementy. Sytém má 13 stupňů volnosti. Autor představuje implementaci řešení dynamických rovnic v Matlabu. K integraci je využíváno Newmarkovy beta metody.

Cílem tohoto odstavce je popsat použité metody, modifikovat představené algoritmy a vyzkoušet efekt redukce na nějakém podobném (větším) systému a zobrazit získané výsledky.

3.3.1 Studovaný systém

Uvažujeme systém bez tlumení

$$M\ddot{u}(t) + R(u(t)) = f(t)$$

doplněný o počáteční podmínky

$$u(t=0) = u_0 \qquad \dot{u}(t=0) = v_0,$$

kde M je matice hmotnosti, u je sloupcový vektor obsahující posunutí jednotlivých vrcholů, R je sloupcový vektor vnitřních sil působících na vrcholy a f je vektor vnějších sil.

Linearizace systému

Takovýto nelineární systém je potřeba diskretizovat v čase a v každém časovém kroku je ještě potřeba iteračně řešit nelineární soustavu rovnic. Jedna možnost, jak se vyhnout iteračnímu řešení nelineární rovnice a zmenšit počet operací, je soustavu linearizovat

$$M\ddot{u}_{i+1} + K_i(u_{i+1} - u_i) = f_{i+1} - R_i,$$

kde

$$K_i = \frac{\partial R}{\partial u}(u_i)$$

je tzv. tečná matice tuhosti. Odpovídá to provedení jedné iterace Newton-Raphsonovy metody. Ukazuje se (viz [9]), že provedení jedné iterace je často dostačující.

Matice redukce

Jak jsme již zmínili, převádí matice redukce U plné souřadnice na redukované a naopak.

$$u(t) = u_0 + Uq(t),$$

kde q jsou redukované souřadnice a u_0 je posunutí, ve kterém jsme se rozhodli soustavu redukovat.

Redukovaná soustava

$$\overline{M}\ddot{q}(t) + \overline{R}(u_0 + Uq(t)) = \overline{f}(t)$$

kde $\overline{M} = U^T M U$, $\overline{R} = U^T R$ a $\overline{f} = U^T f$.

Redukce linearizovaného systému

$$\overline{M}\ddot{q}_{i+1} + \overline{K}_i(q_{i+1} - q_i) = \overline{f}_{i+1} - \overline{R}(u_0 + Uq_i),$$

kde $\overline{K}_i = U^T K_i U$.

3.3.2 Newmark integrační metoda

K integraci je použita tzv. Newmarkova integrační metoda. Nová zrychlení v časovém kroku $i+1$ jsou vyjádřena pomocí ostatních veličin jako

$$\begin{aligned}\ddot{u}_{i+1} &= \frac{1}{\beta\Delta t^2}(u_{i+1} - u_i) - \frac{1}{\beta\Delta t}\dot{u}_i + \left(1 - \frac{1}{2\beta}\right)\ddot{u}_i \\ &= \frac{1}{\beta\Delta t^2}\Delta u_{i+1} - \frac{1}{\beta\Delta t}\dot{u}_i + \left(1 - \frac{1}{2\beta}\right)\ddot{u}_i\end{aligned}$$

Nové rychlosti dostaneme ze vztahu

$$\dot{u}_{i+1} = \dot{u}_i + (1-\gamma)\Delta t\ddot{u}_i + \gamma\Delta t\ddot{u}_{i+1}$$

Parametry integrační metody jsou voleny $\beta = 0.25$ a $\gamma = 0.5$.

Integrace linearizovaného systému Výraz pro zrychlení v čase $i+1$ dosadíme do zlinearizované dynamické rovnice a dostaneme

$$\left(\frac{1}{\beta\Delta t^2}M + K_i\right)\Delta u_{i+1} = f_{i+1} - R_i - M\left(-\frac{1}{\beta\Delta t}\dot{u}_i + \left(1 - \frac{1}{2\beta}\right)\ddot{u}_i\right)$$

Jedná se o lineární soustavu, kterou můžeme vyřešit pro Δu_{i+1}. Nové zrychlení a rychlost dostaneme ze vztahů výše.

Integrace nelineárního systému Jestliže výraz pro zrychlení v čase $i+1$ dosadíme do původní dynamické rovnice, dostaneme nelineární soustavu rovnic

$$\frac{1}{\beta\Delta t^2}M\Delta u_{i+1} + R(u_{i+1}) = f_{i+1} - M\left(-\frac{1}{\beta\Delta t}\dot{u}_i + \left(1 - \frac{1}{2\beta}\right)\ddot{u}_i\right)$$

V tomto případě je potřeba najít řešení u_{i+1} např. pomocí Newton-Raphsonovy metody.

3.3.3 2D tyčový element

2D tyčový element je zřejmě nejjednodušší konečný prvek, pomocí něhož se dá modelovat geometrická nelinearita a lineární materiál. Pomocí tohoto elementu můžeme modelovat tyčové konstrukce spojené klouby, které vykazují velké deformace ale deformace elementu zůstává malá, takže materiál zůstává v rozsahu lineární elasticity. Tyčový element má v referenční konfiguraci délku L_0 a průřez A_0, v aktuální konfiguraci pak délku L a průřez A. Vztah mezi axiální deformací a axiálním napětí (2. Piola-Kirchhoffovo napětí) je specifikován Youngovým modulem pružnosti E (předpokládáme, že v referenční konfiguraci nemá element žádné počáteční napětí S_0).

$$S = E\varepsilon \tag{3.4}$$

Jednotlivý element má 2 vrcholy, každý s x-ovým a y-ovým posunutím, takže má element dohromady 4 posunutí a 4 přidružené síly, které napíšeme pomocí sloupcových vektorů

$$\mathbf{u} = \begin{pmatrix} u_{11} \\ u_{12} \\ u_{21} \\ u_{22} \end{pmatrix} \qquad \mathbf{f} = \begin{pmatrix} f_{11} \\ f_{12} \\ f_{21} \\ f_{22} \end{pmatrix}$$

3.3.4 Dynamická rovnice pro 1 element

V článku [9] je odvozena dynamická rovnice pro 1 tyčový element ve tvaru

$$\frac{1}{2}m\begin{pmatrix} \ddot{u}_{11} \\ \ddot{u}_{12} \\ \ddot{u}_{21} \\ \ddot{u}_{22} \end{pmatrix} = \begin{pmatrix} f_{11} \\ f_{12} \\ f_{21} \\ f_{22} \end{pmatrix} - EA_0\frac{\varepsilon}{\lambda}\begin{pmatrix} -n_{11} \\ -n_{12} \\ n_{11} \\ n_{12} \end{pmatrix},$$

kde $m = \rho_0 A_0 L_0$ je hmotnost elementu, ε je zvolený vztah pro axiální deformaci, na výběr jsou následující vztahy

$$\begin{aligned} \varepsilon &= \lambda - 1 \quad \text{lineární} \\ \varepsilon &= \frac{1}{2}(\lambda^2 - 1) \quad \text{Green-Lagrangeův} \\ \varepsilon &= \ln(\lambda) \quad \text{logaritmický,} \end{aligned}$$

kde $\lambda = \frac{L}{L_0}$ značí relativní prodloužení tyčového elementu a

$$n_{11} = \frac{x}{L} \quad \text{kde} \quad x = x_0 + u_{21} - u_{11}$$
$$n_{12} = \frac{y}{L} \quad \text{kde} \quad y = y_0 + u_{22} - u_{12}$$

x a y jsou x-ová a y-ová délka elementu v aktuální konfiguraci (x_0 a y_0 v referenční). Ve zkrácené maticové notaci by se rovnice pro jeden element dala zapsat takto

$$M_e \ddot{u} = f_e - R_e$$

3.3.5 Chyba redukovaného systému

Pro neredukovaný systém můžeme zavést reziduum

$$Res = f - R - M\ddot{u}$$

Na konci časového kroku můžeme do tohoto rezidua dosadit řešení redukovaného systému a zjistit relativní chybu, která je navržena jako

$$\varepsilon = \frac{\|Res\|}{max(\|f\|,\|R\|,\|M\ddot{u}\|)} \tag{3.5}$$

Pokud chyba přesáhne nastavenou toleranci, můžeme časovou integraci pozastavit a přepočítat matici redukce v aktuální konfiguraci (posunutí).

V případě, že přepočítáme matici redukce, musíme updatovat i redukované rychlosti a redukovaná zrychlení. Ty se z neredukovaných proměnných updatují podle vztahu

$$\dot{q} = \overline{M}^{-1} U^T M \dot{u} \tag{3.6}$$

a stejně pro zrychlení.

Pokaždé, když provedeme takovou aktualizaci, dopouštíme se chyby, která se dá vyjádřit jako (dosazením do $\dot{u} = U\dot{q}$)

$$\dot{u} = U(U^T M U)^{-1} U^T M \dot{u}$$

3.3.6 Implementace v Matlabu

Numerické experimenty provádím v prostředí Matlab pomocí upravených skriptů, které jsou popsané v článku [9]. Skripty jsem modifikoval i na 3D tyčové elementy, abych mohl vyzkoušet 3D tyčovou strukturu. Skripty jsou k nalezení v přiloženém CD. Třeba podotknout, že jsem změnil skoro všechny názvy proměnných v původním kódu, abych se sám v kódu vyznal. Všechny proměnné jsou patřičně zakomentovány pro lepší orientaci případného zájemce. Kromě původních skriptů se dají nalézt funkce na vykreslování, které jsem používal, nebo definice větších tyčových konstrukcí.

Seznam důležitých skriptů

inp.m
Inicializace souřadnic vrcholů, spojení tyčových elementů, materiálových vlastností (Youngův modul pružnosti, hustota tyčového elementu, obsah průřezu v klidovém stavu). Nastavíme počet tečných módů, indexy modálních derivací, které chceme počítat a způsob jejich výpočtu, viz. předchozí text.

init.m
Výpočet matice hmotnosti, inicializace posunutí, rychlostí, zrychlení, vnějších a vnitřních sil.

stiff.m
Vypočítá matici tuhosti K a vektor vnitřních sil R pro aktuální konfiguraci (posunutí u). Posunutí se předá a vypočítá pomocí proměnné uTemp, vypočtená vnitřní síla se předá pomocí proměnné GTemp.

tanmod.m
Vypočítá tečné módy, modální derivace podle zvolené metody a redukovanou matici. Aktualizuje redukované rychlosti a zrychlení podle (3.6).

inter0.m
Počítá neredukovaný systém.

interr.m
Počítá redukovaný systém.

inter0l.m
Počítá neredukovaný linearizovaný systém.

interrl.m
Počítá redukovaný linearizovaný systém.

Důležité datové struktury

crd0
Seznam souřadnic vrcholů ve tvaru

```
[ x,y,z souřadnice 1 vrcholu
  x,y,z souřadnice 2 vrcholu
  x,y,z souřadnice 3 vrcholu
  ... ];
```

conMat
Seznam tyčových elementů ve tvaru

```
[ 1. vrchol, 2. vrchol, číslo komponenty (vždy 1)
  ... ];
```

sndof

Seznam deaktivovaných stupňů volnosti, realizace dirichletových podmínek ve tvaru

```
[ číslo vrcholu, číslo deaktivované souřadnice (x-1, y-2, z-3)
  ... ];
```

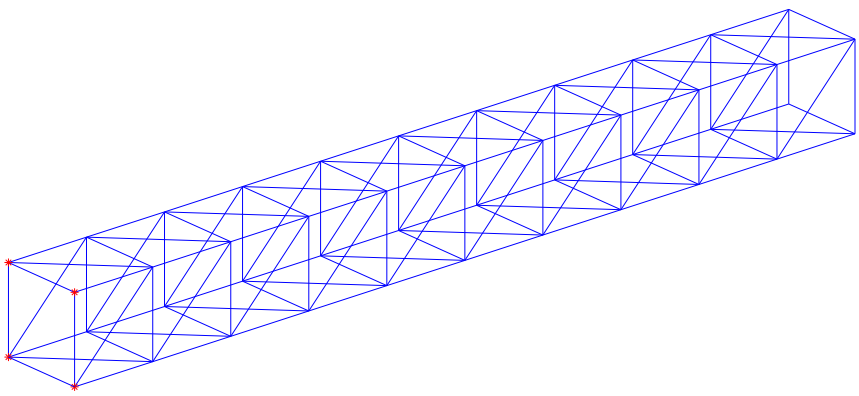
Rád bych experimentálně ověřil následující body

- Tečné módy z modální analýzy dobře popíšou vibrace kolem rovnovážné polohy. U větších deformací selhávají.
- Modální derivace vylepší redukci u tělesa, které podléhá větším deformacím.
- Redukce rovnic zrychluje výpočet na úkor nepřesnosti, které se dopustíme.
- Také bych rád vyzkoušel aposteriorní výpočet matice redukce z více spočítaných deformací (snapshotů) pomocí POD.

Ke všem výpočtům zahrnujícím modální derivace používám 3. nejjednodušší metodu na jejich výpočet. U všech výpočtů je použit Green-Lagrangeův vztah pro axiální deformaci. Matici redukce neaktualizujeme, používáme tedy vypočtené vektory z referenční konfigurace a časově nezávislou matici redukce.

Malé vibrace a větší deformace

Jako příklad jsem vzal 3D tyčovou konstrukci sestavenou ze 44 vrcholů, které jsou spojené dohromady 124 tyčovými elementy. 4 vrcholy na jedné straně jsou napevno přichycené. Celkově má tedy konstrukce $40*3 = 120$ stupňů volnosti (40 volných vrcholů a 3 směry x, y a z). Klidová vzdálenost tyčových elementů je 1m. Používám Green-Lagrangeův vztah pro deformaci. Na obrázku 3.1 je studovaná tyčová konstrukce.

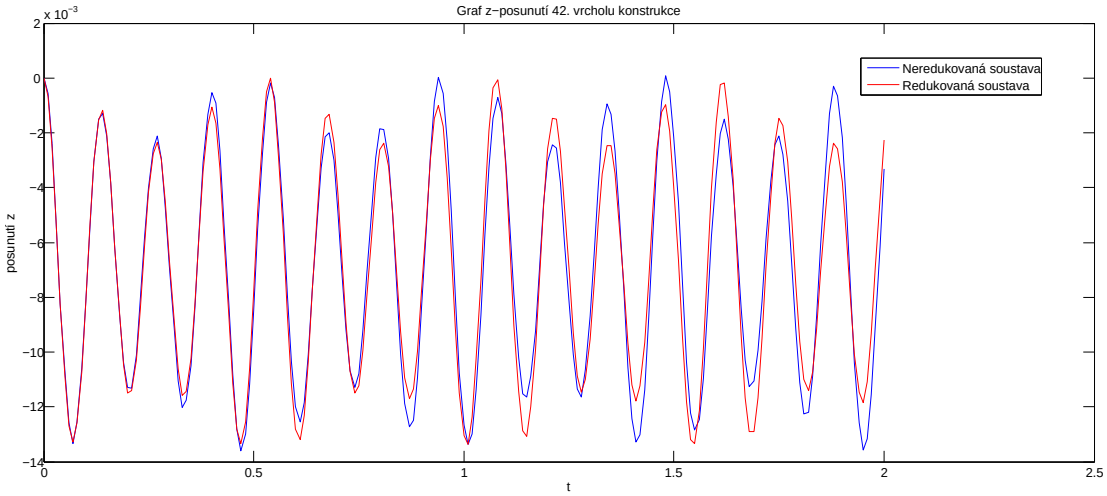


Obrázek 3.1: Studovaná tyčová konstrukce

Červené body značí vrcholy, které jsou napevno přichycené a tvoří Dirichletovy okrajové podmínky. Na těleso nechám působit gravitační sílu a bude se tedy

deformovat vlastní vahou. Změnou Youngova modulu pružnosti E můžeme docílit menších a větších deformací.

Zvolme nejdříve Youngův modul $E = 2.1 \times 10^{11}$, což podle tabulek odpovídá např. oceli. Konstrukce se ohne pouze zhruba o jeden centimetr a bude vibrovat. Na tyto malé deformace se v praxi používá modální analýza. Ukážeme, že redukovaný systém využívající v matici redukce pouze tečné módy vypočtené z modální analýzy tyto malé deformace popisuje dobře. Pro stejně definovanou úlohu vypočteme posunutí pomocí skriptu interr.m, redukovanou matici budou tvořit pouze 3 tečné módy z modální analýzy (tedy počítáme soustavu ve 3 proměnných). Následně v grafu porovnáme časový vývoj posunutí ve směru z u 42. vrcholu (spodní vrchol na kraji, který se vychyluje nejvíce). Porovnání je zobrazeno na obrázku 3.2.



Obrázek 3.2: Porovnání posunutí ve směru z, k redukci použity 3 tečné módy z modální analýzy

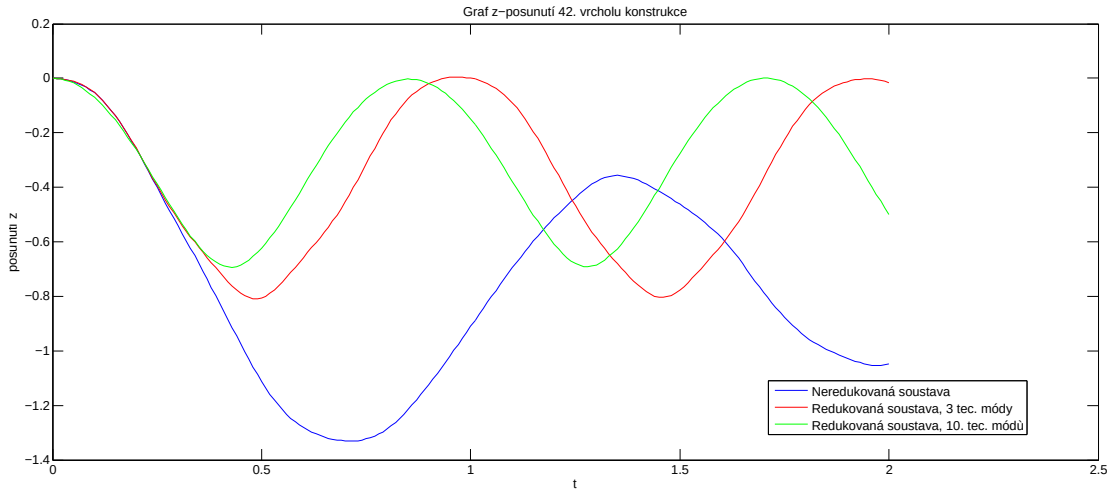
V případě, že zmenšíme Youngův modul pružnosti na $E = 2.1 \times 10^9$, bude konstrukce vykazovat větší deformace (na kraji zhruba 1m). V tomto případě už 3 tečné módy nebudou deformaci popisovat dobře a nepomůže ani jejich zvýšení na např. 10. Na obrázku 3.3 je vidět, že redukce pomocí tečných módů u větší deformace selhává.

Jestliže nyní zkusíme matici redukce nastavit tak, aby obsahovala 4 tečné módy a 6 modálních derivací s indexy (t.j. derivujeme 1. tečný mód podle 1. modální souřadnice, podle druhé modální souřadnice, 2. tečný mód podle 2. modální souřadnice atd.)

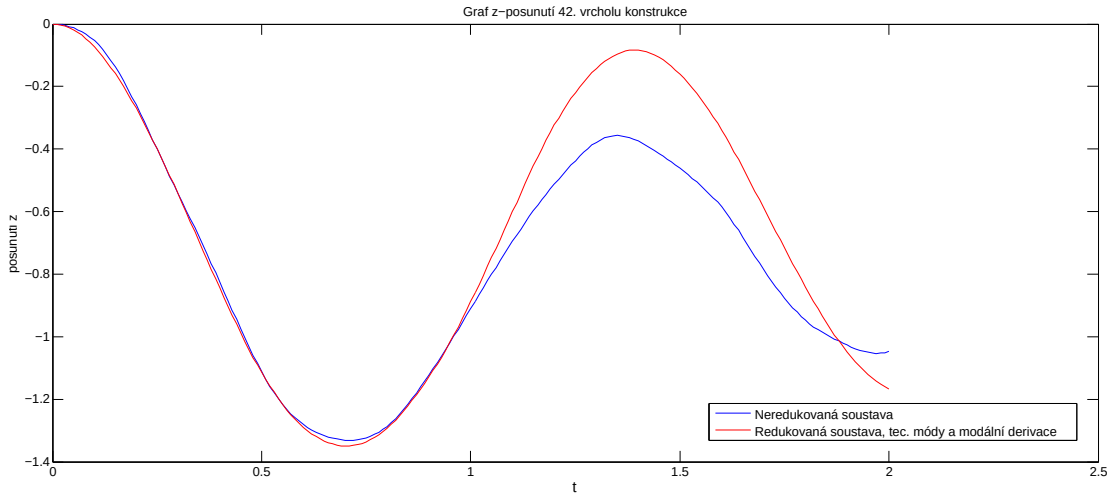
```
modDer = [ 1 1 2 3 3 3
           1 2 2 1 2 3 ];
```

dostaneme výsledek znázorněný na obrázku 3.4.

Je tedy vidět, že pomocí modálních derivací jsme schopni vylepšit redukci, jestliže dochází k větším deformacím, u kterých lineární modální analýza selhává. Výpočet neredukované soustavy trval cca 30 sekund a redukované soustavy zhruba 28 sekund. Pomocí redukce jsme tedy schopni zrychlit výpočty na úkor chyby, které se dopustíme. Tento rozdíl by byl samozřejmě znatelnější u soustav, které mají tisíce nebo statisíce stupňů volnosti. Zde přechod k redukovaným souřadnicím nepředstavuje rapidní zmenšení soustavy.



Obrázek 3.3: Porovnání posunutí ve směru z, k redukci použity jednou 3 tečné módy a podruhé 10 tečných módů

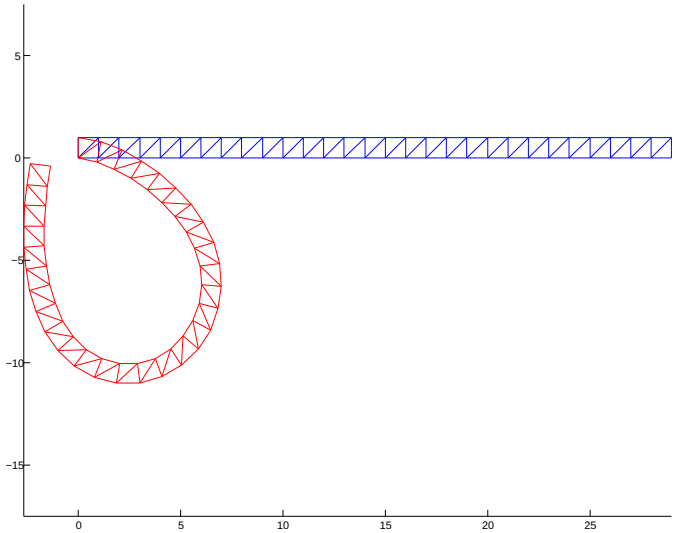


Obrázek 3.4: Porovnání posunutí ve směru z, k redukci použity 4 tečné módy a 6 modálních derivací

Aposteriorní/statistický výpočet matice redukce z nasimulovaných dat

Vyzkouším nyní, jak se dá matice redukce získat aposteriorně pomocí POD. Pro tento účel zkoumám jinou konstrukci, a sice konstrukci ve 2D, která je dost dlouhá (má 60 vrcholů, 117 tyčových elementů, 2 vrcholy vlevo jsou napevno přichycené, systém má 116 stupňů volnosti). Pro změnu vyzkouším skript inter0l, který počítá zlinearizovaný systém, t.j. v každém časovém kroku se provede pouze jedna interace Newton-Raphsonovy metody. Počet časových kroků je nastaven na 400 a jeden časový krok je setina sekundy. Na pravém kraji konstrukce po tuto dobu působí tečná síla, která ji zohne do extrémního tvaru, viz. obrázek 3.5.

V každém časovém kroku uložíme posunutí konstrukce. Dostaneme matici s daty, která je 120×400. Na tuto matici nyní aplikujeme POD a dostaneme vlastní vektory kovariantní matice, ze kterých zkusíme sestavit matici redukce a provést stejnou simulaci s redukovaným systémem. Kód v Matlabu může vypadat např. takto, proměnná „data“ reprezentuje matici s nasimulovanými daty, do matice



Obrázek 3.5: 2D tyčová konstrukce před a po deformaci

PV jsou uloženy vlastní vektory kovariantní matice a do diagonální matice V vlastní čísla.

```
function [PV,V] = POD(data)
% velikost matice s daty
[M,N] = size(data);
% od každého snapshotu odečteme průměr z dat
mn = mean(data,2);
data = data - repmat(mn,1,N);
% vytvoříme kovariantní matici
covariance = 1/N * data * data';
% nalezneme vlastní vektory a vlastní čísla pomocí Matlabovské funkce eig
[PV, V] = eig(covariance);
% extract diagonal of matrix as vector
V = diag(V);
% seřadíme vlastní čísla od největšího po nejmenší a seřadíme i odpovídající vlastní vektory
[junk, sortedIndices] = sort(V,'descend');
V = V(sortedIndices);
PV = PV(:,sortedIndices);
```

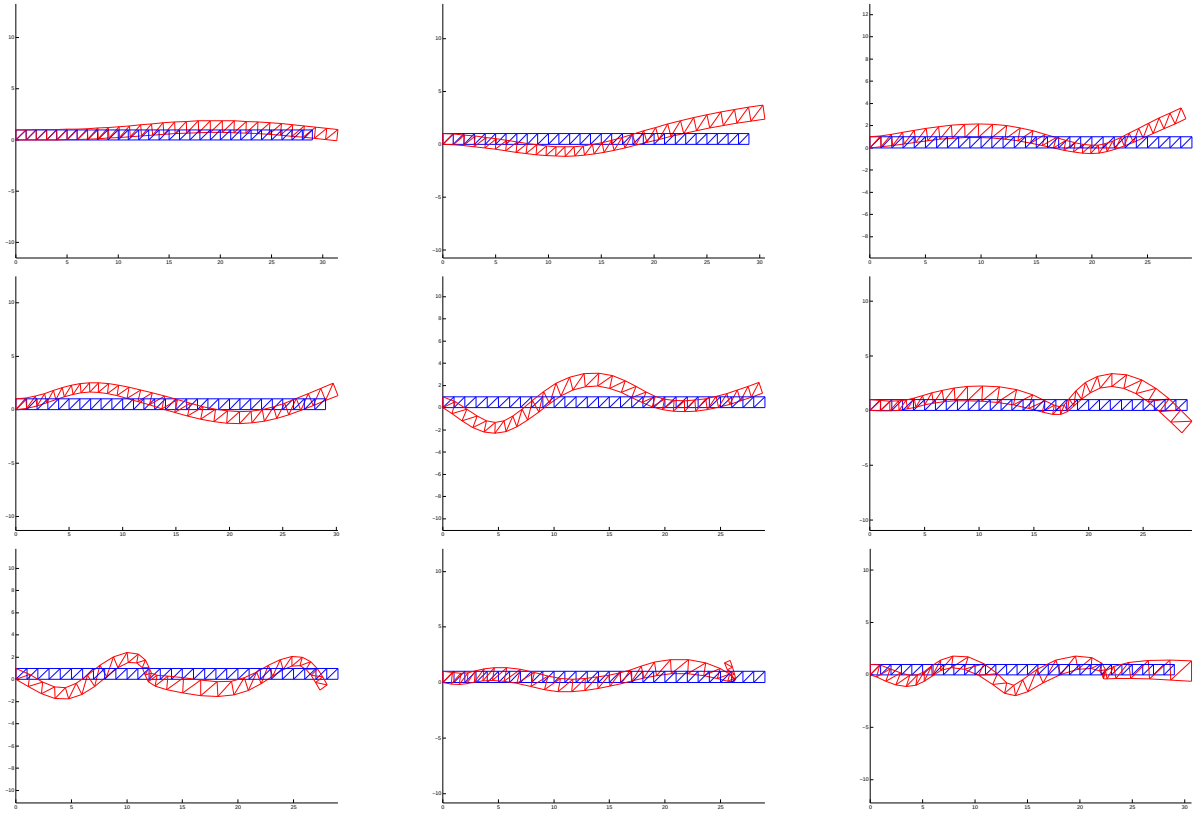
Na základě vztahu mezi počítáním vlastních čísel kovariantní matice a singulárního rozkladu matice dat můžeme vlastní vektory a vlastní čísla získat i ze singulárního rozkladu, jak se to ostatně v praxi často dělá. Kód v Matlabu by vypadal například takto.

```
% vytvoříme matici Y
Y = data' / sqrt(N);
% SVD se o vše postará
[u,S,PV] = svd(Y);
% vlastní čísla jsou singulární čísla na druhou
S = diag(S);
V = S .* S;
```

Zkusme nyní prvních pár vektorů použít do matice redukce a dynamickou úlohu vyřešit znovu pomocí skriptu interrl, tedy redukované zlinearizované soustavy. Jestliže použijeme prvních 5 vektorů, dostaneme úplně jiný výsledek, který

vůbec neodpovídá předchozímu výsledku. Jestliže ale použijeme např. 6. – 10. vektor, dostaneme výsledek velice podobný původnímu, velké ohnutí odpovídá. Jestliže vykreslíme prvních 9 vlastních vektorů, které jsme dostali z POD, je vidět, v čem je zřejmě problém (vektory posunutí vykresluji přenásobené koeficientem 10, aby více vynikl jejich tvar). Až od 6. vektoru vektory „popisují“ také jakési větší ohnutí tyčové kontrukce směrem dolů 3.1. Na obrázku 3.6 je zobrazen časový průběh posunutí z. Je vidět, že při použití prvních 5 vlastních vektorů redukce kompletně selhává. Při použití 8 již však systém popíšeme přesněji.

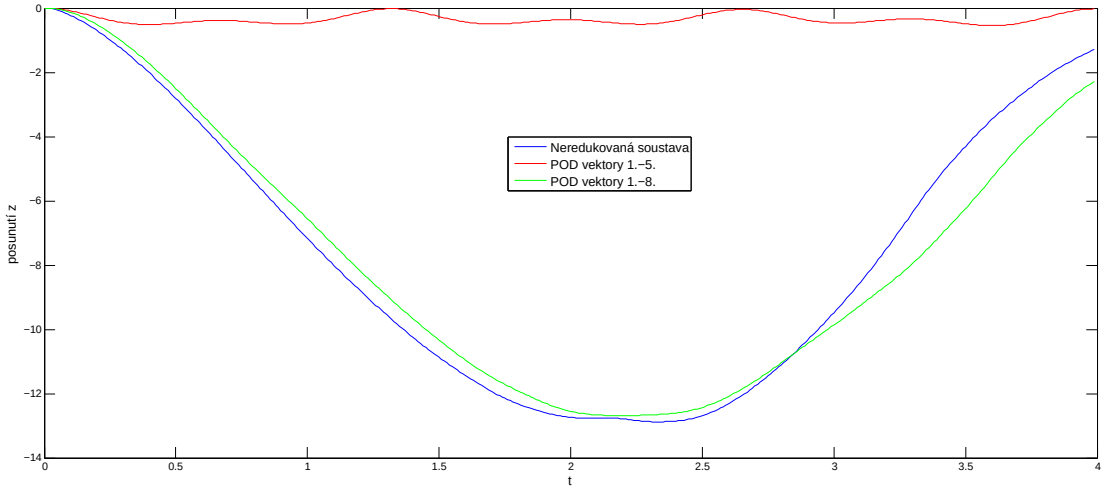
Tabulka 3.1: Vlastní vektory získané z nasimulovaných dat pomocí POD, vektory jsou seřazené od 1. po řádcích



To, že podle tvaru tečných módů a modálních derivací můžeme zvolit redukovanou bázi, demonstruje i článek [4]. Následující obrázky jsou převzaty z tohoto článku. Na obrázku 3.7 jsou zobrazeny některé tečné módy a modální derivace pro těleso připevněné ke stěně. Na druhém obrázku 3.8 jsou vyobrazeny extrémnější deformace objektů, jako např. velké ohnutí lžičky nebo ždímání tělesa. Na základě toho, že víme, jakou deformaci chceme simulovat, můžeme z tvaru tečných módů a modálních derivací vybrat ty, které se na popis budou hodit nejlépe.

Zkusme stejný příklad redukovat i pomocí tečných módů a modálních derivací. Opět modální derivace generujeme 3. způsobem. Redukujeme zlinearizovanou soustavu a výsledky porovnáváme s plným linearizovaným systémem. Do matice redukce používáme 4 tečné módy a postupně 3,4 a 6 modálních derivací, které jsou definované postupně následovně

```
modDer = [ 1 2 2
           1 1 2 ];
```



Obrázek 3.6: Porovnání průběhu posunutí z při redukci pomocí POD

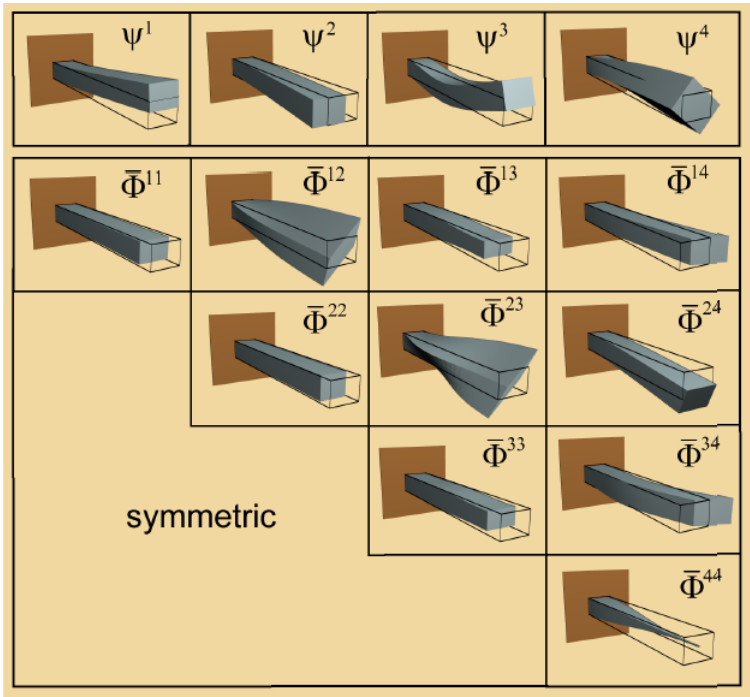
```
modDer = [ 1 2 2 3
           1 1 2 1 ];
```

```
modDer = [ 1 2 2 3 3 3
           1 1 2 1 2 3 ];
```

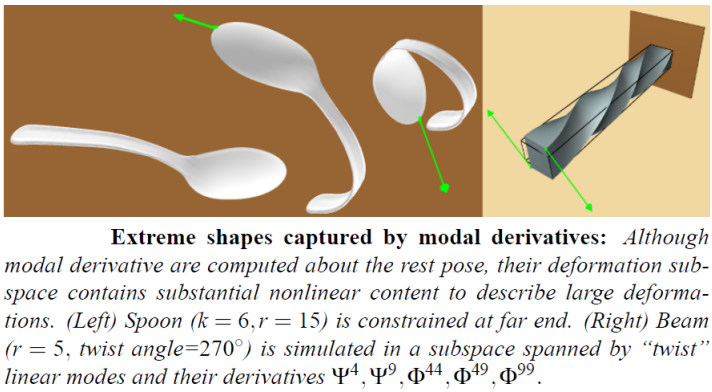
Na obrázku 3.9 je porovnání posunutí z pro poslední vrchol tyčové konstrukce jako v předchozím případu.

Na obrázku 3.10 je konstrukce zaobrazena po deformaci. Maximální chyba redukce (maximum ze všech časových kroků) je 3,4 a 6 modálních derivací postupně 1.2577, 1.1338 a 1.0778.

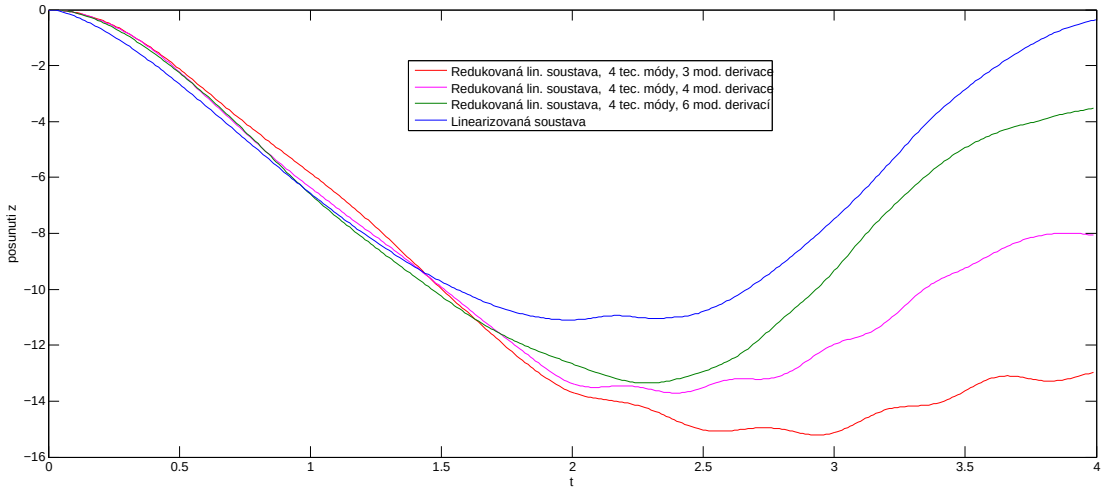
Na začátku jsme zmínili, že třetí možnost na výpočet modálních derivací je nejjednodušší a nejrychlejší. Články [9] a [4] tvrdí, že odlišné generování modálních derivací pomocí těchto tří metod se neprojevuje velkými změnami v redukovaných soustavách. Je proto nejvýhodnější používat metodu nejjednodušší. Zkusme vyzkoušet časovou náročnost výpočtu v prostředí Matlab. Jestliže vytvoříme 2D tyčovou strukturu s 600 stupni volnosti a chceme vypočítat 6 modálních derivací, pak první implementované metodě v Matlabu to trvá zhruba 73 sekund, druhé 42 a třetí 41 sekund.



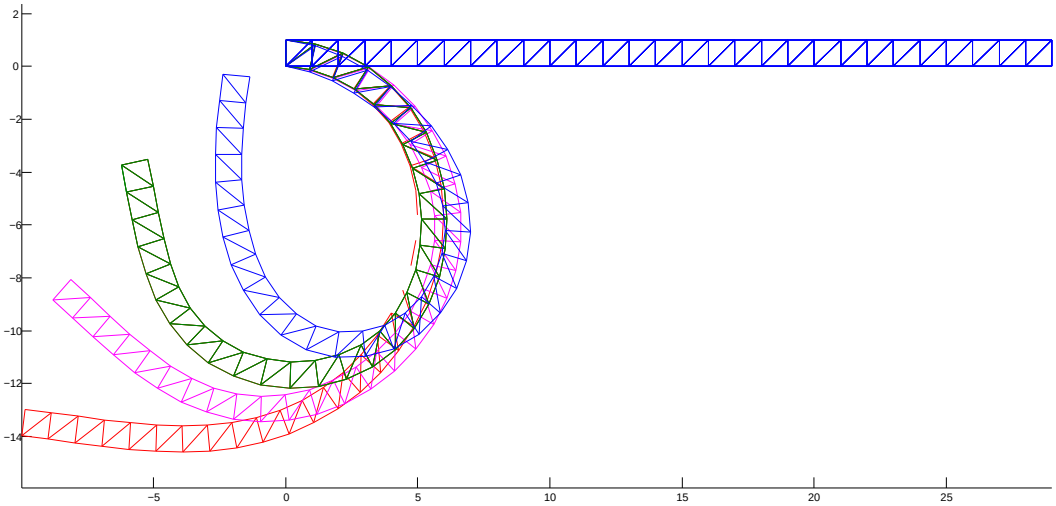
Obrázek 3.7: Dominantní tečné a modální derivace pro objekt přichycený ke stěně. Těleso je modelováno jako St. Venant Kirchoffův materiál



Obrázek 3.8: Předem známe velké deformace můžeme dobře popsat tečnými módy a modálními derivacemi, které podle jejich tvaru vybereme. Zde se jedná o velké ohnutí lžičky a „ždímání“ tělesa, které je přichyceno ke stěně.



Obrázek 3.9: Redukce pomocí modálních derivací, s rostoucím počtem modálních derivací dostaneme přesnější výsledek.



Obrázek 3.10: Deformace v čase 4 s, barvy odpovídají předchozímu obrázku.

Závěr

Metod, které se v počítačové grafice používají na simulace fyzikálních dějů je celkově mnohem více, než jsme ukázali v této práci, a lidé se snaží vyvíjet stále nové a lepší metody. U deformací automobilů se ale zdá, že stále převládají předem vytvořené animace nebo heuristické grafické metody. Tvůrci fyzikálního enginu Rigs of Rods nechali tento projekt jako open source a vyvíjejí jeho nástupce BeamNG, který slaví úspěch a je momentálně zřejmě nejrealističtějším enginem, který simuluje deformace vozidel. Je to právě díky tomu, že se dá auto zdeformovat kompletně a deformace tak vypadá velice realisticky. V jeho jádru se nadále zřejmě skrývá nejen pouze velký počet harmonických oscilátorů, ale i velký počet různých triků a programátorských „hacků“, které zajišťují robustnost celého enginu. Velkým problémem nejen u simulace deformací aut je spojení grafického a fyzikálního modelu.

Korotační lineární metoda konečných prvků ve fyzikálním enginu Digital molecular matter od pixeluxu je zatím jediným případem, kdy se metoda konečných prvků úspěšně implementovala do herního fyzikálního enginu pracujícího v reálném čase. Fyzikální simulace se ale používají např. pro speciální efekty ve filmech, kde se scény mohou počítat i offline. V tomto směru je snaha implementovat stále lepší a více fyzikálnější metody. I v článku [20] autoři krátce hovoří o spojení grafické reprezentace s reprezentací fyzikální. Existuje nějaká velmi hrubá fyzikální síť z tetrahedronů, která je obalená hustší sítí grafických vrcholů. Jejich souřadnice jsou vůči fyzikálním souřadnicím vyjádřeny např. pomocí barycentrických souřadnic vzhledem k vrcholům nejbližšího tetrahedronu.

Specifickým odvětvím zůstávají lékařské operační simulátory, na kterých doktoři trénují různé neinvazivní operace. Zde je totiž potřeba nejen velká rychlost fyzikálního enginu, ale zároveň i přesnost, což představuje velký problém, jelikož tyto dvě vlastnosti jdou proti sobě. Podobnou aplikací je třeba i interaktivní design různých součástek. Zde nachází uplatnění např. právě redukce modelů. Nejen za účelem interaktivního designu byl vyvinut a je stále vyvíjen software Vega FEM. Jeho nasazení, zdá se, ale zůstává stále problematické. U simulátoru s haptickým zařízením, jak se dá nalézt např. v [4] je potřeba odezva nejlépe 1000 Hz, aby se daly dobře modelovat přenášené kontaktní síly. Je tedy potřeba ještě více snímků za sekundu než v počítačových hrách.

Engine Bullet je příklad fyzikálního enginu, ve kterém je implementována jak fyzika pevných těles, tak fyzika deformovatelných těles, která je realizována pomocí poziční dynamiky. Obě metody se dají spojit a používat dohromady. V knihovně je naprogramována detekce kolizí mezi pevnými a deformovatelnými tělesy a jejich interakce. Ukázali jsme, jak se dá velice jednoduše docílit určité heuristické plasticity.

Literatura

[1] D. Baraff. An introduction to physically based modeling: Rigid body simulation ii—nonpenetration constraints. *SIGGRAPH '97 COURSE NOTES*, 1997. http://www.cs.cmu.edu/~baraff/sigcourse/notesd2.pdf.

[2] D. Baraff and A. Witkin. Large steps in cloth simulation. *COMPUTER GRAPHICS Proceedings*, 1998. http://run.usc.edu/cs599-s10/cloth/baraff-witkin98.pdf.

[3] J. Barbič. *Real-time Reduced Large-Deformation Models and Distributed Contact for Computer Graphics and Haptics*. PhD thesis, Carnegie Mellon University, 2007. http://www-bcf.usc.edu/~jbarbic/thesis-barbic.pdf.

[4] J. Barbič and D. James. Real-time subspace integration for st.venant-kirchhoff deformable models. 2005. http://graphics.cs.cmu.edu/projects/stvk/BarbicJames_StVK05.pdf.

[5] J. Bender, K. Erleben, J. Trinkle, and E. Coumans. Interactive simulation of rigid body dynamics in computer graphics. 2012. https://code.google.com/p/bullet/downloads/detail?name=STAR.pdf.

[6] E. Catto. Iterative dynamics with temporal coherence. 2005. http://www.citeulike.org/user/phigley/article/6744456.

[7] E. Catto. Soft constraints, reinventing the spring. 2011. https://code.google.com/p/box2d/downloads/detail?name=GDC2011_Catto_Erin_Soft_Constraints.pdf.

[8] A. Chatterjee. An introduction to the proper orthogonal decomposition. 2000. http://www.iisc.ernet.in/currsci/apr102000/tutorial2.pdf.

[9] J. de Jongh. A reduction method for nonlinear dynamic systems. 1992. http://alexandria.tue.nl/repository/books/629785.pdf.

[10] Z. Dong, S. Liu, Y. Ou, and Y. Zheng. A simple method for real-time metal shell simulation. 2011. http://zhidong.weebly.com/uploads/8/3/7/6/8376282/a_simple_method_for_real-time_metal_shell.pdf.

[11] J.-L. Dulong, F. Druesne, and P. Villon. A model reduction approach for real-time part deformation with nonlinear mechanical behavior. *Springer-Verlag France*, 2007. http://link.springer.com/article/10.1007%2Fs12008-007-0028-y.

[12] D. H. Eberly. *Game physics*. CRC Press, 2010.

[13] H. Garstenauer. A unified framework for rigid body dynamics, 2006. http://www.digitalrune.com/Portals/0/Documents/A%20Unified%20Framework%20for%20Rigid%20Body%20Dynamics.pdf.

[14] E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder. Discrete shells. *Eurographics/SIGGRAPH Symposium on Computer Animation*, 2003. http://www.multires.caltech.edu/pubs/ds.pdf.

[15] T. Jakobsen. Advanced character physics. 2001. http://www.pagines.ma1.upc.edu/~susin/files/AdvancedCharacterPhysics.pdf.

[16] M. Müller. Real time physics, course notes. 2008. http://www.matthiasmueller.info/realtimephysics/coursenotes.pdf.

[17] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable real-time deformations. 2002. http://www.matthiasmueller.info/publications/warp.pdf.

[18] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. 2006. http://www.matthiasmueller.info/publications/posbaseddyn.pdf.

[19] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson. Physically based deformable models in computer graphics. *EUROGRAPHICS 2005*, 2005. http://www.matthiasmueller.info/publications/egstar2005.pdf.

[20] E. G. Parker and J. F. O'Brien. Real-time deformation and fracture in a game environment. 2009. http://graphics.berkeley.edu/papers/Parker-RTD-2009-08/.

[21] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. *Computer Graphics*, 22(4), 1988. http://graphics.stanford.edu/courses/cs448-01-spring/papers/terzopoulos2.pdf.

[22] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics*, 21(4), 1987. http://research.microsoft.com/en-us/um/people/jplatt/siggraph87-elastic.pdf.

[23] M. Wicke, D. Steinemann, and M. Gross. Efficient animation of point-sampled thin shells. 2005. http://graphics.stanford.edu/~wicke/publications/shells/wsg-eaopbts-05.pdf.

Seznam obrázků

Přílohy

Na přiloženém CD se dají nalézt Matlabovské skripty, které jsem používal pro vyzkoušení metody redukce. Jejich stručný popis je ve třetí kapitole. Skripty vytvořil J. de Jongh [9]. Já jsem pro své účely skripty pozměnil, aby se daly spouštět nezávisleji na sobě a změnil jsem názvy skoro všech proměnných, aby více odpovídaly tomu, co reprezentují, a abych kódu porozuměl. Navíc jsou na CD obsaženy skripty, které jsem vytvořil k vykreslování a definici vlastních experimentů s jinými tyčovými strukturami a působícími silami.