České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů

Diplomová práce

Model elektrostatického aktuátoru systému na báze MEMS a jeho simulace

Bc. Jan Plichta

Vedoucí práce: Ing. Michal Štepanovský, Ph.D.

Studijní program: Elektrotechnika a informatika, strukturovaný, Navazující magisterský

Obor: Výpočetní technika

12. května 2014

Poděkování

Rád bych poděkoval vedoucímu mé diplomové práce Ing. Michalu Štepanovskému, Ph. D. za poskytnuté konzultace a své rodině a přátelům za podporu v průběhu tvorby práce. Obzvláště bych rád poděkoval své ženě Petře, za to že věřila i ve chvílích kdy já už ani nedoufal.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 12. 5. 2014 .....

Abstract

In this thesis, the model of an electrostatic actuator is described, implemented and simulated. The model takes into account the interaction between the electrostatic field and mechanical structures of the actuator. It is a distributed parameters model; and is solved by the finite element method. The simulation results are compared with analytical models.

The result of this work is a Java application that simulates the behavior of the electrostatic actuator. The input parameters, e.g. shape and dimensions of mechanical structures, electric potential, etc. are read from input files.

Abstrakt

Cílem této práce je vytvořit model elektrostatického aktuátoru, který bude zohledňovat vzájemnou interakci elektrostatického pole a mechanické struktury. Jedná se o model s rozloženými parametry, který je řešen metodou konečných prvků. S modelem jsou vykonány simulační experimenty a některé z nich jsou porovnány s analytickými modely.

Jedním z výsledků této práce je aplikace v jazyce Java, která modeluje chování elektrostatického aktuátoru. Parametry řešené úlohy, kterými jsou například tvar mechanických struktur či volného prostoru mezi nimi, jsou načítány ze vstupního souboru.

Obsah

1 Úvod **1**

2 Popis problému, specifikace cíle **3**
2.1 Popis problému . . . 3
2.2 Cíl . . . 4

3 Analýza a návrh řešení **5**
3.1 Metoda konečných prvků . . . 5
3.1.1 Slabá formulace úlohy . . . 6
3.1.2 Aproximace hledané funkce . . . 7
3.2 Isoparametrické konečné prvky . . . 8
3.2.1 Integrace nad elementem . . . 12
3.2.1.1 Integrace přes hranici prvku . . . 12
3.2.1.2 Integrace nad šesti uzlovým trojúhelníkový elementem. . . . 13
3.3 Elektrostatika . . . 13
3.3.1 Popis elektrostatického pole . . . 14
3.3.2 Specifikace podmínek a převod na soustavu lineárních rovnic pomocí MKP . . . 14
3.3.3 Programová implementace . . . 15
3.4 Dynamika . . . 16
3.4.1 Popis deformací pružného tělesa . . . 16
3.4.2 Specifikace podmínek a převod na soustavu lineárních rovnic pomocí MKP . . . 17
3.4.3 Programová implementace . . . 20
3.5 Mechanika . . . 20
3.5.1 Specifikace podmínek a převod na soustavu lineárních rovnic pomocí MKP . . . 20
3.6 Deformace konečnoprvkové sítě volného prostoru . . . 21
3.7 Výpočet elektrostatické síly na povrchu elektrody . . . 22

4 Realizace **23**
4.1 Formát vstupních souborů . . . 23
4.1.1 .node . . . 23
4.1.2 .ele . . . 25
4.1.3 .mat . . . 26

4.2 jDistMesh . . . 27
4.2.1 Seznam komponent a popis rozhraní . . . 27
4.2.1.1 Třídy definující vlastnosti sítě z `distmesh.inputs` . . . 27
4.2.1.2 Třídy pracující se sítí z `distmash` . . . 28
4.2.1.3 Přidané třídy . . . 28
4.2.1.4 Pomocné funkce v JavaScriptu . . . 29
4.3 jFEM . . . 29
4.3.1 QuadraticIsoparam . . . 30
4.3.2 QuadraticIsoparamLine . . . 31
4.3.3 ElectrostaticFEM . . . 31
4.3.4 MechanicsFEM . . . 31
4.3.5 DynamicFEM . . . 31
4.3.6 ActuatorFEM . . . 32
4.3.7 Reprezentace sítě . . . 32

5 Testování **35**
5.1 ElektrostaticFEM . . . 35
5.1.1 Testování proti analytickému řešení . . . 35
5.1.1.1 Výpočet analytického řešení . . . 37
5.1.1.2 Porovnání výsledků . . . 37
5.2 MechanicFEM . . . 38
5.2.1 Matice hmotnosti . . . 38
5.2.2 Matice tuhosti . . . 40
5.2.3 Průhyb nosníku zatíženého spojitým zatížením . . . 42
5.3 DynamicFEM . . . 44
5.3.1 Testování proti analytickému řešení . . . 44
5.3.1.1 Podélné kmity tenké tyče . . . 44
5.3.2 Porovnání s programem COMSOL . . . 45
5.3.2.1 Kmitání nosníku namáhaného na tah . . . 45
5.4 ActuatorFEM . . . 47
5.4.1 Kmitání nosníku buzeného elektrostatickým polem . . . 47

6 Závěr **51**

A Seznam použitých zkratek **55**

B Formát vstupních dat **57**
B.1 .node . . . 57
B.2 .ele . . . 57
B.3 .mat . . . 57

C jDistMesh.js **59**

D Instalační a uživatelská příručka **61**
D.1 Minimální požadavky 61
D.2 Instalace obou aplikací 61
D.3 Spuštění a ovládání aplikace jDistMesh 61
D.4 Spuštění a ovládání aplikace jFEM 61
D.4.1 ElectrostaticFEM 62
D.4.2 MechanicsFEM 62
D.4.3 DynamicFEM 62
D.4.4 ActuatorFEM 62

E Obsah přiloženého CD **63**

Seznam obrázků

2.1 Toto zařízení může být použito jako mikrosenzor, stejně jako mikroaktuátor. [2] 4

3.1 Triangulace oblasti Ω. 5
3.2 Jak jsou číslovány uzly ve vstupním souboru v rámci jednoho trojúhelníku . . 8
3.3 Značení pro výpočet intenzity na hraně 22

4.1 Jak jsou číslovány uzly ve vstupním souboru v rámci jednoho trojúhelníku . . 26

5.1 Schéma zadání pro výpočet rozložení potenciálu 35
5.2 Síť pro testování výpočtu rozložení elektrostatického potenciálu 36
5.3 Závislost velikosti chyby numerického řešení na velikosti kroku sítě. 39
5.4 Poloha uzlů v trojúhelníku $e1$ 40
5.5 Poloha uzlů v trojúhelníku $e2$ 40
5.6 Poloha uzlů v trojúhelníku $e3$ 41
5.7 Nosník zatížený spojitým zatížením 43
5.8 Naznačení struktury sítě pro výpočet vetknutého nosníku 43
5.9 Kmitání tyče ve zvoleném módu 46
5.10 Schéma zadání nosníku namáhaného na tah 47
5.11 Kmitání nosníku namáhaného na tah 48
5.12 Schéma zadání nosníku namáhaného elektrostatickou silou 48
5.13 Kmitání nosníku namáhaného elektrostatickou silou 49

Seznam tabulek

3.1 Gaussovy body pro integraci v 1D 13
3.2 Gaussovy body pro integraci v 2D 14

5.1 Závislost chyby numerického řešení na průměrné délce strany trojúhelníku. . 38
5.2 Závislost velikosti chyby numerického řešení na délce dílku ve směru osy x. . 43

E.1 Obsah přiloženého CD 63

Seznam ukázek kódu

4.1 Universální funkce pro výpočet integrálu 30
4.2 Výpočet integrálu podle vzorce (3.79) 30
5.1 Skript na vygenerování sítě pro testování `ElektrostaticFEM` 35
C.1 Pomocné funkce, konstanty a struktury v JavaScriptu 59

Kapitola 1

Úvod

Ve své bakalářské práci [15] jsem se zabýval výpočtem rozložení elektrostatického potenciálu pomocí metody sítí. Jelikož mě toto téma zaujalo, chtěl jsem se na něj navázat ve své diplomové práci. Proto jsem si vybral téma Model elektrostatického aktuátoru systému na báze MEMS a jeho simulace.

Systémy na bázi MEMS jsou velmi malá elektromechanická zařízení, jejichž velikost se může pohybovat již v řádech mikrometrů. Aktuátory jsou spolu se senzory a jinými typy systému (filtry, oscilátory) jednou ze součástí MEMS systémů. Převádí elektrickou energii na energii pohybovou. Ve své práci se budu zabývat aktuátory elektrostatickými tedy aktuátory, které převádí energii elektrostatického pole na energii mechanickou.

V této práci bude provedena analýza elektrostatického aktuátoru se záměrem vytvořit model s rozloženými parametry. Bude popsáno chování elektrostatického pole, mechanických struktur a vzájemná interakce mezi nimi. Bude navrženo jak jednotlivé části modelovat pomocí metody konečných prvků a jak je mezi sebou propojit do jednoho systému, který bude zohledňovat jak mechanické vlastnosti tak i elektrostatické vlastnosti celého systému. Metoda konečných prvků potřebuje mít oblast výpočtu rozdělenou na elementy. Těmito elementy budou pro potřeby této práce šestiuzlové trojúhelníky. Pro rozdělení oblasti výpočtu - vytvoření sítě - bude vytvořen pomocný program na generování těchto sítí.

Jedním z výsledků této práce bude aplikace v jazyce Java, která bude schopná modelovat libovolně zadaný elektrostatický aktuátor. Parametry budou načítány ze vstupních souborů. Vstupní soubory obsahují materiálové vlastnosti, strukturu sítě (tedy geometrický popis zadání) a okrajové a počáteční podmínky např. elektrostatický potenciál v jednotlivých částech systému nebo ukotvení systému k okolí.

Jednotlivé subsystémy programu budou otestovány a porovnány s analytickým řešením, případně s řešením z jiných modelačních programů. Na závěr bude otestováno propojení jednotlivých komponent systému do jednoho celku a provedena simulace aktuátoru.

Kapitola 2

Popis problému, specifikace cíle

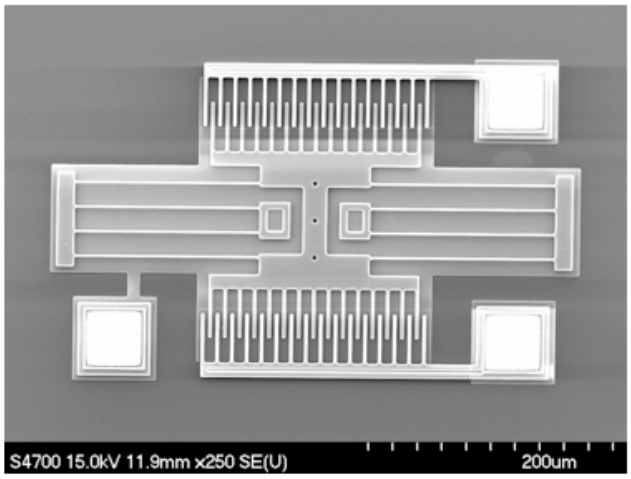
2.1 Popis problému

Aktuátor je podle [12] definován jako funkční prvek, který spojuje informační proces s technickým nebo netechnickým (např. biologickým) procesem Aktuátory se používají pro převod elektrické energie v jiné fyzikální veličiny (mechanické vlnění, tlak, teplotu, pohyb). Jsou tudíž opakem senzorů.

Podle způsobu přeměny energie lze aktuátory rozdělit na elektromagnetické, elektrochemické, piezoelektrické, opto-elektrické, magnetostrikční atd. S těmito aktuátory se setkáváme v běžném životě – například u motoru, jehož rotor je uveden do pohybu elektromagnetickým polem; reproduktoru, jehož membrána se pohybuje v rytmu elektrického kmitání a tím vytváří akustický tlak; dále se jedná o diody ve skenerech, dotykových displejích, či optických myších atd. V mé diplomové práci se budu zabývat MEMS (mikro-elektro-mechanical-systems) aktuátory. Tedy aktuátory, které převádí elektrickou energii na energii pohybovou. Samotný pojem MEMS zahrnuje miniaturní mechanická a elektromechanická zařízení, jejichž velikost se může pohybovat od jednoho mikronu až po několik milimetrů. I přes tuto velikost provádí tato zařízení mnohem větší mechanické výkony než by se z této jejich velikosti dalo usuzovat. Typy mechanických MEMS zařízení se kromě své velikosti liší i svou strukturou, od relativně jednoduchých, které nemají žádné pohyblivé prvky až po velmi složité elektromechanické systémy s více pohyblivými částmi. Ukázka mikroaktátoru je zobrazena na obrázku 2.1

Součástí MEMS jsou senzory, aktuátory, ale také oscilátory či filtry. MEMS mají široké využití. V běžném životě se s nimi setkáváme například u mobilních telefonů, které reagují na pohyb, tabletů, jejichž obrazovka se otočí, pokud pohybujeme přístrojem, airbagů, které vystřelí při extrémním zpomalení auta nebo u mikromechanických pump dávkujících léky.

Elektrostatické mikroaktuátory využívají podle [11] přitahování mezi opačně nabitými nebo kapacitně se chovajícími elektrodami k pohybu nebo deformaci tělesa. Mají nízkou spotřebu, rychlou odezvu a jejich výroba je nenáročná. Jejich ovládání je složitější díky nelineárnímu chování.



Obrázek 2.1: Toto zařízení může být použito jako mikrosenzor, stejně jako mikroaktuátor. [2]

2.2 Cíl

Cílem této práce je navrhnout model elektrostatického aktuátoru.

Za tímto účelem bude provedena analýza z fyzikálního pohledu a vytvořen matematický model popisující chování elektrostatického aktuátoru. Navržený model bude zohledňovat vzájemnou interakci mechanické struktury a elektrostatického pole. Bude se jednat o model s rozloženými parametry a bude navrženo jeho řešení pomocí metody konečných prvků.

Následně bude podle této analýzy naimplementován program v jazyce Java. Budou popsány jednotlivé komponenty programu a jejich vzájemná interakce. Také bude navržena struktura vstupních souborů a vytvořen pomocný program na generování trojúhelníkové sítě.

Pro ověření funkčnosti vytvořeného programu budou provedeny testy jednotlivých funkčních částí a programu jako celku.

Kapitola 3

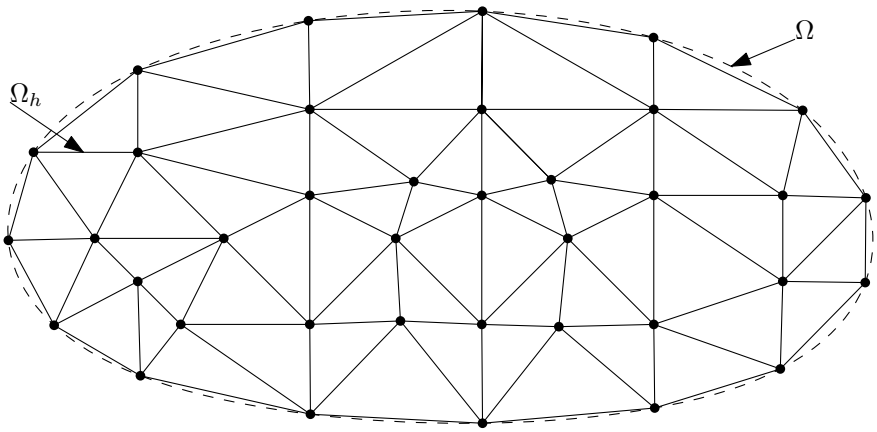
Analýza a návrh řešení

V této části bude popsán z fyzikálního pohledu elektrostatický aktuátor a budou navrženy metody jeho výpočtu.

Elektrostatický aktuátor je složen z mechanických částí a z volného prostoru mezi nimi. Pro modelování mechanických částí bude použit model lineárního elastického tělesa bez detekce kolizí. Volný prostor budeme modelovat jako oblast, která neklade odpor pro pohyb mechanických částí. Nad celou oblastí bude spočteno rozložení potenciálu a z něj síla působící na mechanické části.

Dále budou stručně nastíněny principy použitých metod.

3.1 Metoda konečných prvků



Obrázek 3.1: Triangulace oblasti Ω.

Při použití metody konečných prvků aproximujeme oblast Ω oblastí Ω_h tak, že jí pokryjeme nepřekrývajícími se trojúhelníky (viz obrázek 3.1). Oblast Ω je znázorněna čárkovaně, oblast Ω_h je znázorněna plnou čarou a vrcholy triangulace jsou znázorněny plnými kružnicemi. Funkci u aproximujeme lineární kombinací funkcí e_i. Tato funkce e_i je rovna nule ve všech

vrcholech triangulace kromě vrcholu i, kde je rovna jedné. Zaměříme se na variantu metody, kde funkce e_i jsou v každém trojúhelníku lineární.

3.1.1 Slabá formulace úlohy

Pro metodu konečných prvků budeme potřebovat takzvanou slabou formulaci úlohy. Pro ilustraci slabé formulace vyjdeme z klasického zadání okrajové úlohy dle Laplaceovy diferenciální rovnice 3.1. Čili hledáme funkci u a chceme, aby platilo:

$$\Delta u \ = 0 \tag{3.1}$$

pro všechny vnitřní body oblasti Ω, kde Δ je Laplaceův diferenciální operátor. Zároveň chceme, aby funkce u byla na hranicích oblasti Ω rovna funkci g, tedy aby platilo

$$u = g \tag{3.2}$$

na Γ, který označuje hranici oblasti Ω.

Rovnici $\Delta u \ = 0$ vynásobíme hladkou funkcí v, která je rovna nule na hranici Γ. Tím získáme

$$v \ \Delta u \ = 0 \quad \text{v } \Omega$$

Rovnice bude platit pro každou spojitou funkci v. Protože výraz je roven nule v každém bodě oblasti, bude i integrál levé strany přes oblast Ω roven nule. Tedy platí následující vztah

$$\int_\Omega v \ \Delta u \ \mathrm{d}x = 0. \tag{3.3}$$

Za předpokladu existence všech tří integrálů můžeme použít Greenovy věty

$$\int_\Omega v \ \Delta u \ \mathrm{d}x = -\int_\Omega \nabla v \ .\nabla u \ \mathrm{d}x + \int_\Gamma v \ \frac{\partial u}{\partial \mathbf{n}} \mathrm{d}s,$$

převedeme rovnici (3.3) na

$$\int_\Omega \nabla v \ \cdot \nabla u \ \mathrm{d}x = \int_\Gamma v \ \frac{\partial u}{\partial \mathbf{n}} \mathrm{d}s, \tag{3.4}$$

kde Γ je hranice oblasti Ω, $\mathbf{n}$ je normálový vektor na $\mathrm{d}s$, $\frac{\partial u}{\partial \mathbf{n}}$ je derivace ve směru normálového vektoru a operace $\cdot$ je skalární součin. Protože v je rovna nule na hranici oblasti, člen na pravé straně je nulový. Slabým řešením úlohy (3.1), (3.2) se nazývá funkce, která je dostatečně hladká, splňuje okrajové podmínky[1] a vyhovuje rovnosti

$$\int_\Omega \nabla v \ \cdot \nabla u \ \mathrm{d}x = 0$$

pro každou dostatečně spojitou funkci v, nulovou na hranici Γ.[2]

Lze ukázat, že každé klasické řešení problému (3.1), (3.2) je také slabé řešení. Slabému řešení však mohou vyhovovat i funkce, které nejsou klasickým řešením (např. nemají omezené druhé derivace).

[1] Zde stačí pouze integrovatelnost s druhým kvadrátem včetně prvních derivací funkce a splnění okrajových podmínek ve smyslu stop, viz např. [13]

[2] Zde opět požadujeme integrovatelnost s kvadrátem včetně jejích prvních derivací, t.j. $v \in W_2^1(\Omega)$, a nulovost na hranici Γ ve smyslu stop.

3.1.2 Aproximace hledané funkce

Oblast Ω rozdělíme na trojúhelníky například tak, jak je vidět na obrázku 3.1. Rozdělení oblasti na trojúhelníky neboli triangulace musí splňovat podmínku, že žádný vrchol trojúhelníku nesmí ležet na hraně jiného trojúhelníku. Původní oblast Ω nahradíme sjednocením Ω_h všech m trojúhelníku K_i, tedy

$$\Omega_h = \bigcup_{i=1}^{m} K_i.$$

Koeficient triangulace h udává velikost nejdelší hrany v celé triangulaci, neboli vyjádřeno vztahem

$$h = \max_{i=1,\ldots,m} \left(\max\{a_i, b_i, c_i\}\right),$$

kde a_i, b_i a c_i jsou délky hran trojúhelníka K_i.

Pomocí této triangulace zadefinujeme konečný prostor funkcí V_h nad oblastí Ω_h. Funkce patřící do prostoru V_h budou spojité nad celou oblastí Ω_h, na hranici oblasti budou rovny nule a budou lineární nad každým trojúhelníkem K_i.

Nechť $\{x_j \mid j = 1, \ldots, n\}$ je množina všech vrcholů triangulace, kde n je počet vrcholů triangulace. Dále si definujme množinu spojitých a zároveň po částech lineárních funkcí e_j. Každá funkce e_j bude svázaná s bodem x_j podle následujícího předpisu

$$e_j(x_i) = \begin{cases} 1 & \text{pokud } x_i = x_j \\ 0 & \text{pokud } x_i \neq x_j \end{cases}$$

a bude lineární nad každým trojúhelníkem K_i. Takto definovaná množina funkcí e_j tvoří bázi prostoru V_h.

Aproximaci funkce u budeme označovat symbolem $\tilde{u}$ a budeme ji hledat v prostoru funkcí V_h. Každá funkce z tohoto prostoru se dá vyjádřit jako lineární kombinace prvků báze e_i prostoru. Tedy i funkce $\tilde{u}$ se dá vyjádřit vztahem

$$\tilde{u} = \sum_{i=1}^{n} u_i e_i, \tag{3.5}$$

kde u_i jsou reálná čísla.

Vztah (3.5) si vyjádříme pomocí funkcí z prostoru V_h. Za u dosadíme vztah (3.5) a za v všechny bázové funkce e_j, které jsou rovny nule na Γ_h. Tím získáme následující soustavu rovnic

$$\int_{\Omega_h} \left(\sum_{i=1}^{n} u_i \nabla e_i \right) . \nabla e_j \, \mathrm{d}x = 0.$$

Soustavu rovnic doplníme o aproximaci okrajové podmínky. Přidáme

$$u_i = g(x_i)$$

pro všechny $x_i \in \Gamma_h$.

Vytknutím sumy před integrál získáme vztah

$$\sum_{i=1}^{n} \left(u_i \int_{\Omega_h} \nabla e_i . \nabla e_j \, \mathrm{d}x \right) = 0. \tag{3.6}$$

Tento vztah představuje soustavu lineárních rovnic. Neznámé soustavy jsou u_i. Matice soustavy bude mít koeficienty

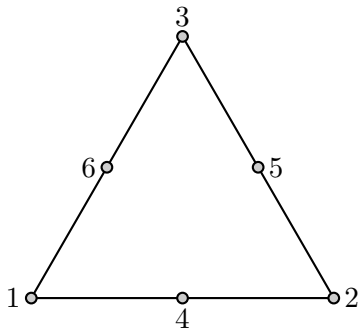
$$a_{ij} = \int_{\Omega_h} \nabla e_i . \nabla e_j \, \mathrm{d}x$$

a bude velmi řídká, neboť integrál bude různý od nuly jen pro ty funkce, které odpovídají vrcholům x_i a x_j, které spojuje hrana triangulace.

3.2 Isoparametrické konečné prvky

Popis isoparametrického trojúhelníku vnikl na základě konzultací s vedoucím práce.

Kvadratický isoparametrický trojúhelník je určen šesti uzlovými body (uzly). Jejich rozmístění na trojúhelníku je na znázorněno na obrázku 3.2.



Obrázek 3.2: Jak jsou číslovány uzly ve vstupním souboru v rámci jednoho trojúhelníku

Oblast Ω_e isoparametrického kvadratického trojúhelníkového elementu je popsána rovnicemi

$$x(\xi_1, \xi_2, \xi_3) = \sum_{a=1}^{6} N_a(\xi_1, \xi_2, \xi_3) \widetilde{x}_a, \tag{3.7}$$

$$y(\xi_1, \xi_2, \xi_3) = \sum_{a=1}^{6} N_a(\xi_1, \xi_2, \xi_3) \widetilde{y}_a, \tag{3.8}$$

kde $\widetilde{x}_a$ je x-ová souřadnice a-tého uzlu, $\widetilde{y}_a$ je y-ová souřadnice a-tého uzlu a N_a jsou tzv. tvarové funkce, pro které platí

$$N_1 = \xi_1(2\xi_1 - 1), \tag{3.9}$$
$$N_2 = \xi_2(2\xi_2 - 1), \tag{3.10}$$
$$N_3 = \xi_3(2\xi_3 - 1), \tag{3.11}$$
$$N_4 = 4\xi_1\xi_2, \tag{3.12}$$
$$N_5 = 4\xi_2\xi_3, \tag{3.13}$$
$$N_6 = 4\xi_3\xi_1, \tag{3.14}$$

a $\xi_i \in \langle 0, 1\rangle$, $i = 1, 2, 3$ jsou barycentrické souřadnice, mezi kterými platí vztah

$$\xi_1 + \xi_2 + \xi_3 = 1. \tag{3.15}$$

Rovnice (3.7) a (3.8) představují transformační vztah mezi kartézskými souřadnicemi x, y a barycentrickými souřadnicemi ξ_1, ξ_2, ξ_3. Dále budeme potřebovat vyjádřit inverzní vztah $\xi_i = f(x, y)$. Přímočará možnost jak toho docílit je ze soustavy rovnic (3.7), (3.8) a (3.15) přímo vyjádřit $\xi_i(x, y)$, cože je sice technicky možné, ale poměrně náročné. Transformační vztah proto raději vyjádříme pouze lokálně pro nějaký bod (ξ_1, ξ_2, ξ_3) pomocí totálních diferenciálů (tj. linearizací) vztahů (3.7) a (3.8)

$$\mathrm{d}x = \frac{\partial x}{\partial \xi_1}\,\mathrm{d}\xi_1 + \frac{\partial x}{\partial \xi_2}\,\mathrm{d}\xi_2 + \frac{\partial x}{\partial \xi_3}\,\mathrm{d}\xi_3, \tag{3.16}$$
$$\mathrm{d}y = \frac{\partial y}{\partial \xi_1}\,\mathrm{d}\xi_1 + \frac{\partial y}{\partial \xi_2}\,\mathrm{d}\xi_2 + \frac{\partial y}{\partial \xi_3}\,\mathrm{d}\xi_3, \tag{3.17}$$
$$\tag{3.18}$$

kde parciální derivace kartézských složek podle barycentrických souřadnic získáme snadno jako

$$\frac{\partial x}{\partial \xi_i} = \sum_{a=1}^{6} \frac{\partial N_a}{\partial \xi_i}\widetilde{x}_a, \tag{3.19}$$
$$\frac{\partial y}{\partial \xi_i} = \sum_{a=1}^{6} \frac{\partial N_a}{\partial \xi_i}\widetilde{y}_a, \tag{3.20}$$

kde parciální derivace tvarových funkcí podle ξ_i, které mají tvar

$$\frac{\partial \mathbf{N}}{\partial \xi_1} = \begin{pmatrix} 4\xi_1 - 1 \\ 0 \\ 0 \\ 4\xi_2 \\ 0 \\ 4\xi_3 \end{pmatrix}, \quad \frac{\partial \mathbf{N}}{\partial \xi_2} = \begin{pmatrix} 0 \\ 4\xi_2 - 1 \\ 0 \\ 4\xi_1 \\ 4\xi_3 \\ 0 \end{pmatrix}, \quad \frac{\partial \mathbf{N}}{\partial \xi_3} = \begin{pmatrix} 0 \\ 0 \\ 4\xi_3 - 1 \\ 0 \\ 4\xi_2 \\ 4\xi_1 \end{pmatrix}. \tag{3.21}$$

Rovnice (3.16)–(3.17) představují lineární soustavu dvou rovnic pro tři neznámé. Zbývající rovnici získáme diferencováním rovnice (3.15)

$$\mathrm{d}\xi_1 + \mathrm{d}\xi_2 + \mathrm{d}\xi_3 = 0. \tag{3.22}$$

Soustavu rovnic (3.16), (3.17) a (3.22) můžeme zapsat v maticové notaci jako

$$\begin{pmatrix} \mathrm{d}x \\ \mathrm{d}y \\ 0 \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial x}{\partial \xi_2} & \frac{\partial x}{\partial \xi_3} \\ \frac{\partial y}{\partial \xi_1} & \frac{\partial y}{\partial \xi_2} & \frac{\partial y}{\partial \xi_3} \\ 1 & 1 & 1 \end{pmatrix}}_{\mathbf{J}} \begin{pmatrix} \mathrm{d}\xi_1 \\ \mathrm{d}\xi_2 \\ \mathrm{d}\xi_3 \end{pmatrix} \tag{3.23}$$

a odtud konečně můžeme vyjádřit žádaný transformační vztah

$$\begin{pmatrix} \mathrm{d}\xi_1 \\ \mathrm{d}\xi_2 \\ \mathrm{d}\xi_3 \end{pmatrix} = \mathbf{J}^{-1} \begin{pmatrix} \mathrm{d}x \\ \mathrm{d}y \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{\partial \xi_1}{\partial x} & \frac{\partial \xi_1}{\partial y} & \bar{J}_{13} \\ \frac{\partial \xi_2}{\partial x} & \frac{\partial \xi_2}{\partial y} & \bar{J}_{23} \\ \frac{\partial \xi_3}{\partial x} & \frac{\partial \xi_3}{\partial y} & \bar{J}_{33} \end{pmatrix} \begin{pmatrix} \mathrm{d}x \\ \mathrm{d}y \\ 0 \end{pmatrix}. \tag{3.24}$$

Uvažujme nyní nějaké skalární pole, jehož aproximaci nad oblastí Ω_e můžeme psát ve tvaru

$$u(x,y) = \sum_{a=1}^{6} N_a\left(\xi_1(x,y), \xi_2(x,y), \xi_3(x,y)\right) \widetilde{u}_a, \tag{3.25}$$

kde $\widetilde{u}_a$ jsou hodnoty pole v uzlových bodech oblasti Ω_e. Vyjádříme si derivaci této funkce podle kartézských souřadnic, pro které za pomocí derivování složené funkce můžeme psát

$$\frac{\partial u}{\partial x} = \frac{\partial u}{\partial \xi_1}\frac{\partial \xi_1}{\partial x} + \frac{\partial u}{\partial \xi_2}\frac{\partial \xi_2}{\partial x} + \frac{\partial u}{\partial \xi_3}\frac{\partial \xi_3}{\partial x}, \tag{3.26}$$

$$\frac{\partial u}{\partial y} = \frac{\partial u}{\partial \xi_1}\frac{\partial \xi_1}{\partial y} + \frac{\partial u}{\partial \xi_2}\frac{\partial \xi_2}{\partial y} + \frac{\partial u}{\partial \xi_3}\frac{\partial \xi_3}{\partial y} \tag{3.27}$$

a v maticové notaci

$$\begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{pmatrix} = \begin{pmatrix} \frac{\partial \xi_1}{\partial x} & \frac{\partial \xi_2}{\partial x} & \frac{\partial \xi_3}{\partial x} \\ \frac{\partial \xi_1}{\partial y} & \frac{\partial \xi_2}{\partial y} & \frac{\partial \xi_3}{\partial y} \end{pmatrix} \begin{pmatrix} \frac{\partial u}{\partial \xi_1} \\ \frac{\partial u}{\partial \xi_2} \\ \frac{\partial u}{\partial \xi_3} \end{pmatrix}, \tag{3.28}$$

kde parciální derivace pole u podle barycentrických souřadnic snadno dostaneme z (3.25) jako

$$\frac{\partial u}{\partial \xi_i} = \sum_{a=1}^{6} \frac{\partial N_a}{\partial \xi_i} \widetilde{u}_a, \tag{3.29}$$

takže (3.28) můžeme rozepsat jako

$$\frac{\partial u}{\partial x} = \sum_{i=1}^{3}\sum_{a=1}^{6} \frac{\partial \xi_i}{\partial x}\frac{\partial N_a}{\partial \xi_i}\widetilde{u}_a = \sum_{a=1}^{6} \frac{\partial N_a}{\partial x}\widetilde{u}_a \tag{3.30}$$

$$\frac{\partial u}{\partial y} = \sum_{i=1}^{3}\sum_{a=1}^{6} \frac{\partial \xi_i}{\partial y}\frac{\partial N_a}{\partial \xi_i}\widetilde{u}_a = \sum_{a=1}^{6} \frac{\partial N_a}{\partial y}\widetilde{u}_a \tag{3.31}$$

Neznámé parciální derivace barycentrických souřadnic ξ_i podle kartézských souřadnic x, y potřebné v (3.28) získáme jako prvky matice $\mathbf{J}^{-1}$ (viz. (3.24)). Tuto inverzi můžeme provést buď numericky, což není příliš efektivní, uvážíme-li, že se tento výpočet opakuje pro každý integrační bod každého elementu. Nebo můžeme inverzi matice $\mathbf{J}$ vypočítat explicitně pomocí determinantů a subdeterminantů

$$\mathbf{J}^{-1} = \frac{1}{\det \mathbf{J}} \begin{bmatrix} (-1)^2 \begin{vmatrix} \frac{\partial y}{\partial \xi_2} & \frac{\partial y}{\partial \xi_3} \\ 1 & 1 \end{vmatrix} & (-1)^3 \begin{vmatrix} \frac{\partial x}{\partial \xi_2} & \frac{\partial x}{\partial \xi_3} \\ 1 & 1 \end{vmatrix} & (-1)^4 \begin{vmatrix} \frac{\partial x}{\partial \xi_2} & \frac{\partial x}{\partial \xi_3} \\ \frac{\partial y}{\partial \xi_2} & \frac{\partial y}{\partial \xi_3} \end{vmatrix} \\ (-1)^3 \begin{vmatrix} \frac{\partial y}{\partial \xi_1} & \frac{\partial y}{\partial \xi_3} \\ 1 & 1 \end{vmatrix} & (-1)^4 \begin{vmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial x}{\partial \xi_3} \\ 1 & 1 \end{vmatrix} & (-1)^5 \begin{vmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial x}{\partial \xi_3} \\ \frac{\partial y}{\partial \xi_1} & \frac{\partial y}{\partial \xi_3} \end{vmatrix} \\ (-1)^4 \begin{vmatrix} \frac{\partial y}{\partial \xi_1} & \frac{\partial y}{\partial \xi_2} \\ 1 & 1 \end{vmatrix} & (-1)^5 \begin{vmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial x}{\partial \xi_2} \\ 1 & 1 \end{vmatrix} & (-1)^6 \begin{vmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial x}{\partial \xi_2} \\ \frac{\partial y}{\partial \xi_1} & \frac{\partial y}{\partial \xi_2} \end{vmatrix} \end{bmatrix} \tag{3.32}$$

$$\mathbf{J}^{-1} = \begin{bmatrix} \frac{\partial \xi_1}{\partial x} & \frac{\partial \xi_1}{\partial y} & \bar{J}_{13} \\ \frac{\partial \xi_2}{\partial x} & \frac{\partial \xi_2}{\partial y} & \bar{J}_{23} \\ \frac{\partial \xi_3}{\partial x} & \frac{\partial \xi_3}{\partial y} & \bar{J}_{33} \end{bmatrix} = \frac{1}{\det \mathbf{J}} \begin{bmatrix} \begin{vmatrix} \frac{\partial y}{\partial \xi_2} & \frac{\partial y}{\partial \xi_3} \\ 1 & 1 \end{vmatrix} & -\begin{vmatrix} \frac{\partial x}{\partial \xi_2} & \frac{\partial x}{\partial \xi_3} \\ 1 & 1 \end{vmatrix} & \begin{vmatrix} \frac{\partial x}{\partial \xi_2} & \frac{\partial x}{\partial \xi_3} \\ \frac{\partial y}{\partial \xi_2} & \frac{\partial y}{\partial \xi_3} \end{vmatrix} \\ -\begin{vmatrix} \frac{\partial y}{\partial \xi_1} & \frac{\partial y}{\partial \xi_3} \\ 1 & 1 \end{vmatrix} & \begin{vmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial x}{\partial \xi_3} \\ 1 & 1 \end{vmatrix} & -\begin{vmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial x}{\partial \xi_3} \\ \frac{\partial y}{\partial \xi_1} & \frac{\partial y}{\partial \xi_3} \end{vmatrix} \\ \begin{vmatrix} \frac{\partial y}{\partial \xi_1} & \frac{\partial y}{\partial \xi_2} \\ 1 & 1 \end{vmatrix} & -\begin{vmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial x}{\partial \xi_2} \\ 1 & 1 \end{vmatrix} & \begin{vmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial x}{\partial \xi_2} \\ \frac{\partial y}{\partial \xi_1} & \frac{\partial y}{\partial \xi_2} \end{vmatrix} \end{bmatrix} \tag{3.33}$$

$$\frac{\partial \xi_1}{\partial x} = \frac{1}{\det \mathbf{J}} \begin{vmatrix} \frac{\partial y}{\partial \xi_2} & \frac{\partial y}{\partial \xi_3} \\ 1 & 1 \end{vmatrix} = \frac{1}{\det \mathbf{J}} \left(\frac{\partial y}{\partial \xi_2} - \frac{\partial y}{\partial \xi_3} \right) = \frac{J_{y23}}{\det \mathbf{J}} \tag{3.34}$$

$$\frac{\partial \xi_2}{\partial x} = \frac{-1}{\det \mathbf{J}} \begin{vmatrix} \frac{\partial y}{\partial \xi_1} & \frac{\partial y}{\partial \xi_3} \\ 1 & 1 \end{vmatrix} = \frac{1}{\det \mathbf{J}} \left(\frac{\partial y}{\partial \xi_3} - \frac{\partial y}{\partial \xi_1} \right) = \frac{J_{y31}}{\det \mathbf{J}} \tag{3.35}$$

$$\frac{\partial \xi_3}{\partial x} = \frac{1}{\det \mathbf{J}} \begin{vmatrix} \frac{\partial y}{\partial \xi_1} & \frac{\partial y}{\partial \xi_2} \\ 1 & 1 \end{vmatrix} = \frac{1}{\det \mathbf{J}} \left(\frac{\partial y}{\partial \xi_1} - \frac{\partial y}{\partial \xi_2} \right) = \frac{J_{y12}}{\det \mathbf{J}} \tag{3.36}$$

$$\frac{\partial \xi_1}{\partial y} = \frac{-1}{\det \mathbf{J}} \begin{vmatrix} \frac{\partial x}{\partial \xi_2} & \frac{\partial x}{\partial \xi_3} \\ 1 & 1 \end{vmatrix} = \frac{1}{\det \mathbf{J}} \left(\frac{\partial x}{\partial \xi_3} - \frac{\partial x}{\partial \xi_2} \right) = \frac{J_{x32}}{\det \mathbf{J}} \tag{3.37}$$

$$\frac{\partial \xi_2}{\partial y} = \frac{1}{\det \mathbf{J}} \begin{vmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial x}{\partial \xi_3} \\ 1 & 1 \end{vmatrix} = \frac{1}{\det \mathbf{J}} \left(\frac{\partial x}{\partial \xi_1} - \frac{\partial x}{\partial \xi_3} \right) = \frac{J_{x13}}{\det \mathbf{J}} \tag{3.38}$$

$$\frac{\partial \xi_3}{\partial y} = \frac{-1}{\det \mathbf{J}} \begin{vmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial x}{\partial \xi_2} \\ 1 & 1 \end{vmatrix} = \frac{1}{\det \mathbf{J}} \left(\frac{\partial x}{\partial \xi_2} - \frac{\partial x}{\partial \xi_1} \right) = \frac{J_{x21}}{\det \mathbf{J}} \tag{3.39}$$

kde jsme zavedli

$$J_{xij} = \left(\frac{\partial x}{\partial \xi_i} - \frac{\partial x}{\partial \xi_j} \right), \qquad J_{yij} = \left(\frac{\partial y}{\partial \xi_i} - \frac{\partial y}{\partial \xi_j} \right), \qquad i, j = 1, 2, 3. \tag{3.40}$$

Determinant Jacobiho matice je

$$
\begin{aligned}
\det \mathbf{J} &= \begin{vmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial x}{\partial \xi_2} & \frac{\partial x}{\partial \xi_3} \\ \frac{\partial y}{\partial \xi_1} & \frac{\partial y}{\partial \xi_2} & \frac{\partial y}{\partial \xi_3} \\ 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} J_{x1} & J_{x2} & J_{x3} \\ J_{y1} & J_{y2} & J_{y3} \\ 1 & 1 & 1 \end{vmatrix} = \\
&= J_{x1}(J_{y2} - J_{y3}) + J_{x2}(J_{y3} - J_{y1}) + J_{x3}(J_{y1} - J_{y2}) = \\
&= J_{x1}J_{y23} + J_{x2}J_{y31} + J_{x3}J_{y12} = \\
&= J_{x1}J_{y23} + J_{x2}J_{y31} + \underbrace{(J_{x1}J_{y31} - J_{x1}J_{y31})}_{0} + J_{x3}J_{y12} + \underbrace{(J_{x1}J_{y12} - J_{x1}J_{y12})}_{0} = \\
&= J_{x21}J_{y31} - J_{x13}J_{y12} + \underbrace{J_{x1}J_{y23} + J_{x1}J_{y31} + J_{x1}J_{y12}}_{0} = \\
&= J_{x21}J_{y31} - J_{x13}J_{y12}
\end{aligned}
\tag{3.41}
$$

Konečně pro parciální derivace tvarových funkcí podle kartézských souřadnic můžeme psát

$$
\frac{\partial N_a}{\partial x} = \frac{1}{\det \mathbf{J}} \left(J_{y23} \frac{\partial N_a}{\partial \xi_1} + J_{y31} \frac{\partial N_a}{\partial \xi_2} + J_{y12} \frac{\partial N_a}{\partial \xi_3} \right), \tag{3.42}
$$

$$
\frac{\partial N_a}{\partial y} = \frac{1}{\det \mathbf{J}} \left(J_{x32} \frac{\partial N_a}{\partial \xi_1} + J_{x13} \frac{\partial N_a}{\partial \xi_2} + J_{x21} \frac{\partial N_a}{\partial \xi_3} \right). \tag{3.43}
$$

3.2.1 Integrace nad elementem

3.2.1.1 Integrace přes hranici prvku

Na hranicích elementu jsou použity bázové funkce

$$
\mathbf{N}^T = \begin{pmatrix} N_1 \\ N_2 \\ N_3 \end{pmatrix} = \begin{pmatrix} \xi_1(2\xi_1 - 1) \\ \xi_2(2\xi_2 - 1) \\ 4\xi_1\xi_2 \end{pmatrix} \tag{3.44}
$$

Pro vektorovou veličinu jako je např. síla budeme používat matici bázových funkci ve tvaru

$$
\mathbf{N_u} = \begin{pmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 \end{pmatrix} \tag{3.45}
$$

Budeme je potřebovat jen na výpočet integrálu (3.81)

$$
\int_{\Gamma^e} h \, \mathbf{N}_p^T \mathbf{f}_B \, \mathrm{d}\Gamma^e, \tag{3.46}
$$

kde $\mathbf{f}_B = \mathbf{N}\widetilde{\mathbf{f}}_B$.

Při převodu tohoto křivkového integrálu na obyčejný integrál je potřeba vyjádřit výraz

$$
\mathrm{d}\Gamma^e = \sqrt{\phi'(t)^2 + \psi'(t)^2} \, \mathrm{d}t,
$$

w_i	t_i
0.1184634425281	0.046910077030
0.2393143352497	0.230765344947
0.2844444444444	0.500000000000
0.2393143352497	0.769234655053
0.1184634425281	0.953089922969

Tabulka 3.1: Gaussovy body pro integraci v 1D

kde $\phi(t) = \mathbf{N}(t, 1-t)\widetilde{\mathbf{x}}$ a $\psi(t) = \mathbf{N}(t, 1-t)\widetilde{\mathbf{y}}$. Pro výpočet derivace $\phi'(t)$ a $\psi'(t)$ je potřeba určit

$$\frac{\partial \mathbf{N}(t, 1-t)}{\partial t} = \frac{\partial \mathbf{N}}{\partial \xi_1}(t, 1-t) - \frac{\partial \mathbf{N}}{\partial \xi_2}(t, 1-t) \tag{3.47}$$

Parciální derivace bázových funkcí podle ξ_i mají následující tvar

$$\frac{\partial \mathbf{N}}{\partial \xi_1} = \begin{pmatrix} 4\xi_1 - 1 \\ 0 \\ 4\xi_2 \end{pmatrix}, \qquad \frac{\partial \mathbf{N}}{\partial \xi_2} = \begin{pmatrix} 0 \\ 4\xi_2 - 1 \\ 4\xi_1 \end{pmatrix}. \tag{3.48}$$

Pro vyčíslení integrálu použijeme Gaussovu kvadraturní formuli [**?**]

$$\int_0^1 f(t)\,\mathrm{d}t \approx \sum_{i=0}^{n} w_i f(t_i). \tag{3.49}$$

Podle této formule aproximuje integrál (3.46) výrazem

$$\int_{\Gamma^e} h\, \mathbf{N}_p^T \mathbf{f}_B\,\mathrm{d}\Gamma^e \approx \sum_{i=0}^{n} w_i h\, \mathbf{N}_p^T(t_i, 1-t_i)\mathbf{f}_B(t_i, 1-t_i) J_\Gamma(t_i, 1-t_i), \tag{3.50}$$

kde $J_\Gamma(t_i, 1-t_i) = J_\Gamma(t_i) = \sqrt{\phi'(t)^2 + \psi'(t)^2}$. Pro $n = 4$ jsou váhy w_i a hodnoty t_i pro integraci přes hranici uvedeny v tabulce 3.1.

3.2.1.2 Integrace nad šesti uzlovým trojúhelníkový elementem.

Pro vyčíslení integrálu použijeme modifikované Gaussovu kvadraturní formule pro trojúhelníkový element

$$\int_{\Omega^e} f(\xi_1, \xi_2, \xi_3)\,\mathrm{d}\Omega^e \approx \sum_{i=0}^{n} w_i f(\xi_1^i, \xi_2^i, \xi_3^i) J_\Omega(\xi_1^i, \xi_2^i, \xi_3^i), \tag{3.51}$$

kde $J_\Omega = \frac{1}{2}\det \mathbf{J}$ a $\det \mathbf{J}$ se spočítá podle vzorce (3.41). Při výpočtu budeme používat devět bodů tj. $n = 8$. Hodnoty jsou w_i a $\xi_1^i, \xi_2^i, \xi_3^i$ jsou shrnuty do tabulky 3.2.

3.3 Elektrostatika

Odvození základních vztahů pro výpočet elektrostatického pole pomocí MKP vniklo na základě konzultací s vedoucím práce.

w_i	ξ_1^i	ξ_2^i	ξ_3^i
0.205950504760887	0.43752524838339	0.12494950323323	0.43752524838338
0.205950504760887	0.43752524838339	0.43752524838338	0.12494950323323
0.205950504760887	0.12494950323324	0.43752524838338	0.43752524838338
0.063691414286223	0.03747742075009	0.79711265186007	0.16540992738984
0.063691414286223	0.16540992738984	0.79711265186007	0.03747742075009
0.063691414286223	0.03747742075009	0.16540992738984	0.79711265186007
0.063691414286223	0.79711265186007	0.16540992738984	0.03747742075009
0.063691414286223	0.16540992738984	0.03747742075009	0.79711265186007
0.063691414286223	0.79711265186007	0.03747742075009	0.16540992738984

Tabulka 3.2: Gaussovy body pro integraci v 2D

3.3.1 Popis elektrostatického pole

Základní rovnice pro popis elektrostatického pole odvodíme z Maxwellových rovnic:

$$\nabla \times \mathbf{E} = 0, \tag{3.52}$$

$$\nabla^T \mathbf{D} = q_e, \tag{3.53}$$

$$\mathbf{D} = \varepsilon \mathbf{E}, \tag{3.54}$$

kde $\mathbf{E}$ vektor intenzity elektrického pole, $\mathbf{D}$ vektor elektrické indukce a ε permitivita prostředí. Rotace intenzity elektrického pole $\mathbf{E}$ je nulová a lze ji tudíž vyjádřit jako gradient skalární veličiny — elektrického potenciálu V_e.

$$\mathbf{E} = -\nabla V_e \tag{3.55}$$

Z rovnic (3.53), (3.54) a (3.55) odvodíme rovnici

$$-\nabla^T \varepsilon \nabla V_e = q_e \tag{3.56}$$

popisující elektrostatické pole v prostředí charakterizovaném materiálovou veličinou ε.

3.3.2 Specifikace podmínek a převod na soustavu lineárních rovnic pomocí MKP

Je dána oblast Ω s rozložením náboje q_e a permitivitou ε. Hledáme rozložení elektrického potenciálu V_e v dané oblasti. Intenzitu elektrostatického pole $\mathbf{E}$ lze následně vypočítat pomocí rovnice (3.55). Specifikace podmínek:
Jsou dány funkce:

$$q_e : \Omega \to \Re$$
$$\varepsilon : \Omega \to \Re$$

Hledáme funkci:

$$V_e : \Omega_\Gamma \to \Re$$

Na celé oblasti platí rovnice (3.56), tj.:

$$-\nabla^T \varepsilon \nabla V_e = q_e$$

Okrajové Dirichletovy a Neumannovy podmínky:

$$\begin{aligned} V_e &= V_0 \quad \text{na}\, \Gamma_d \\ \frac{\partial V_e}{\partial n} &= 0 \quad \text{na}\, \Gamma_n \end{aligned}$$

kde $\Gamma_d \cup \Gamma_n = \Gamma$ a zároveň $\Gamma_d \cap \Gamma_n = \emptyset$ a symbol Γ označuje hranici oblasti Ω. Symbol Ω_Γ je definován jako $\Omega_\Gamma = \Omega \cup \Gamma$.

Rovnici (3.56) převedeme na tzv. slabou formulaci úlohy, čímž se sníží řád diferenciální rovnice a zároveň aplikujeme Neumannovy podmínky.

$$\int_\Omega \varepsilon (\nabla w)^T \nabla V_e \,\mathrm{d}\Omega + \int_\Omega w q_e \,\mathrm{d}\Omega - \int_{\Gamma_n} w \frac{\partial V_e}{\partial n} \,\mathrm{d}\Gamma = 0 \tag{3.57}$$

kde w je váhová funkce. Podle MKP aproximujeme rozložení elektrického potenciálu nad každým elementem. Jako váhové funkce si zvolíme samotné tvarové funkce. A protože Neumannovy podmínky jsou nulové, lze rovnici (3.57) převést do diskrétního tvaru:

$$\sum_{a=1}^{n_{eq}} \sum_{b=1}^{n_{eq}} \left(\int_\Omega \varepsilon (\nabla N_a)^T \nabla N_b \,\mathrm{d}\Omega \, \widetilde{V}_{eb} + \int_\Omega N_a q_e \,\mathrm{d}\Omega \right) = 0, \tag{3.58}$$

Tuto rovnici můžeme vyjádřit v maticové tvaru takto:

$$\mathbf{K}_{V_e} \widetilde{\mathbf{V}}_e = \mathbf{f}_{V_e} \tag{3.59}$$

kde

$$\mathbf{K}_{V_e} = \coprod_{e=1}^{n_e} \mathbf{k}_{V_e}^e, \quad \mathbf{k}_{V_e}^e = [k_{pq}], \quad k_{pq} = \int_{\Omega^e} \varepsilon (\nabla N_p)^T \nabla N_q \,\mathrm{d}\Omega^e \tag{3.60}$$

$$\mathbf{f}_{V_e} = \coprod_{e=1}^{n_e} \mathbf{f}_{V_e}^e, \quad \mathbf{f}_{V_e}^e = [f_p], \quad f_p = \int_{\Omega^e} N_p q_e \,\mathrm{d}\Omega^e \tag{3.61}$$

kde n_{eq} je počet rovnic, symbol $\coprod$ označuje operátor skládání, n_e je počet elementů, N (N_a, N_b, N_p, N_q) jsou zvolené tvarové funkce. Operace skládání přičítá hodnoty prvků dílčích matic k odpovídajícím prvkům výsledné matice.

3.3.3 Programová implementace

Pro výpočet elektrostatického pole je potřeba sestavit matici $\mathbf{K}_{V_e}$ a vektor pravých stran $\mathbf{f}_{V_e}$. Ty sestavíme podle rovnic (3.60), (3.61). K tomu je potřeba vyčíslit uvedené integrály a dílčí matice a vektory poskládat do výsledného tvaru.

V oblasti Ω nejsou žádné zdroje elektrostatického pole, proto bude vektor $\mathbf{f}_{V_e} = \mathbf{0}$. Následně aplikujeme Dirichletovu okrajovou podmínku, která doplní na patřičných pozicích zadanou hodnotu elektrostatického potenciálu.

Způsob úpravy soustavy $\mathbf{K}_{V_e}\widetilde{\mathbf{V}}_e = \mathbf{f}_{V_e}$ aplikací Dirichletovy okrajové podmínky bychom mohli popsat následovně: Máme-li v uzlu s indexem i zadanou hodnotu V_0, odečteme i-tý sloupce matice $\mathbf{K}_{V_e}$ vynásobený hodnotou V_0 od vektoru $\mathbf{f}_{V_e}$, pak i-tý sloupce a i-tý řádek vynulujeme a na i-tou pozici na diagonále matice dosadíme jedničku. Nakonec na i-tou pozici vektoru $\mathbf{f}_{V_e}$ dosadíme hodnotu V_0.

3.4 Dynamika

Odvození základních vztahů pro výpočet deformací lineárního pružného tělesa pomocí MKP vniklo na základě konzultací s vedoucím práce.

3.4.1 Popis deformací pružného tělesa

Pro odvození rovnic popisujících deformaci pružného tělesa začneme u Cauchyho tenzoru napětí. Ten udává stav napětí v libovolném bodě tělesa, které je v rovnovážném stavu a působí na něj vnější povrchové a objemové síly.

$$\boldsymbol{\sigma} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{pmatrix}, \quad \text{kde } \sigma_{ij} = \sigma_{ji} \tag{3.62}$$

Deformace v libovolném bodě můžeme popsat pomocí tenzoru deformace

$$\boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_{xx} & \varepsilon_{xy} \\ \varepsilon_{yx} & \varepsilon_{yy} \end{pmatrix}, \quad \text{kde } \varepsilon_{ij} = \varepsilon_{ji} \tag{3.63}$$

Nech libovolný bod P nezatíženého tělesa má souřadnice x_p a y_p. Působením vnějších sil se tento bod posune do bodu P' o souřadnicích x'_p a y'_p. Vektor posunutí $\mathbf{u}$ je dán rozdílem nových a původních souřadnic a má souřadnice $u_x = x'_p - x_p$ a $u_y = y'_p - y_p$.

Chování lineárního elastického materiálu můžeme popsat pomocí tří tenzorových rovnic. První rovnice je pohybová rovnice – druhý Newtonův zákon:

$$\nabla^T \boldsymbol{\sigma}_i + f_i = \rho_M \frac{\partial^2 u_i}{\partial t^2} \quad \text{pro } i = x, y \tag{3.64}$$

kde $\boldsymbol{\sigma} = (\sigma_{xi}\sigma_{yi})^T$, f_i jsou vnější síly, ρ_M je hustota materiálu a u_i je posunutí.

Druhou rovnicí pro lineární materiál je rovnice určující tři deformační veličiny ε_{ij}, neboť $\varepsilon_{ij} = \varepsilon_{ji}$:

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial j} + \frac{\partial u_j}{\partial i}\right) \quad \text{pro } i, j = x, y. \tag{3.65}$$

Třetí rovnicí je zákon pružnosti udávající vztah mezi napětím a deformací. Hookov zákon pružnosti pro lineární izotropní materiál můžeme napsat ve tvaru:

$$\begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{pmatrix} = \begin{pmatrix} \lambda_L + 2\mu_L & \lambda_L & 0 \\ \lambda_L & \lambda_L + 2\mu_L & 0 \\ 0 & 0 & \mu_L \end{pmatrix} \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ 2\varepsilon_{xy} \end{pmatrix}, \tag{3.66}$$

kde λ_L a μ_L jsou Laméovi koeficienty pro 2D podle [5] a [7]:

$$\lambda_L = \frac{\nu E}{1-\nu^2} \quad \text{a} \quad \mu_L = \frac{E}{2(1+\nu)} \tag{3.67}$$

kde E je Youngův modul pružnosti a ν je Poissonova konstanta.

Tím máme sestavený systém 8 určující 8 neznámých ($\sigma_{ij}, \varepsilon_{ij}, u_i$ pro $i, j \in \{x, y\}$). Z těchto rovnic můžeme postupným substituováním formulovat vektorovou Navier-Cauchyho diferenciální rovnici pro posunutí:

$$\mu_L(\nabla^2)^T\mathbf{u} + (\lambda_L + \mu_L)\nabla(\nabla^T\mathbf{u}) + \mathbf{f} = \rho\frac{\partial^2\mathbf{u}}{\partial t^2}. \tag{3.68}$$

Protože Cauchyho tenzor napětí stejně jako tenzor deformace jsou symetrické a je výhodné je zapisovat jako 3-složkový vektor podle Voigtovy notace:

$$\boldsymbol{\sigma}_{Voigt} = \begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{pmatrix} = \begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{pmatrix}, \quad \boldsymbol{\varepsilon}_{Voigt} = \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ 2\varepsilon_{xy} \end{pmatrix} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{pmatrix}. \tag{3.69}$$

Pomocí teto notace přepíšeme Hookův zákon (3.66) do tvaru:

$$\boldsymbol{\sigma}_{Voigt} = \mathbf{E}\boldsymbol{\varepsilon}_{Voigt} \tag{3.70}$$

kde matice $\mathbf{E}$ je matice v rovnici (3.66).

Zadefinujeme si diferenciální operátor $\boldsymbol{\beta}$ následujícím způsobem:

$$\boldsymbol{\beta} = \begin{pmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{pmatrix}. \tag{3.71}$$

Pomocí tohoto operátoru můžeme přepsat rovnici (3.65) na tvar:

$$\boldsymbol{\varepsilon}_{Voigt} = \boldsymbol{\beta}\mathbf{u}. \tag{3.72}$$

Když dosadíme rovnice (3.70) a (3.71) do rovnice (3.64), dostaneme

$$\boldsymbol{\beta}^T\mathbf{E}\boldsymbol{\beta}\mathbf{u} + \mathbf{f} = \rho\frac{\partial^2\mathbf{u}}{\partial t^2}, \tag{3.73}$$

která je pouze jiným vyjádřením Navier-Cauchyho diferenciální rovnici. Z této rovnice budeme vycházet při výpočtu mechanického namáhání a deformací tělesa vlivem vnějších sil.

3.4.2 Specifikace podmínek a převod na soustavu lineárních rovnic pomocí MKP

Jsou dány funkce:

$$\begin{aligned} \mathbf{u}_0 &: \Omega \to \Re^2 \\ \dot{\mathbf{u}}_0 &: \Omega \to \Re^2 \\ \rho, E, \nu &: \Omega \to \Re \\ \mathbf{f} &: \Omega \to \Re^2 \end{aligned}$$

Hledáme funkci:

$$\mathbf{u}(t) : \Omega_\Gamma \times [0, T] \to \Re^2$$

Na celé oblasti platí rovnice (3.73):

$$\boldsymbol{\beta}^T \mathbf{E} \boldsymbol{\beta} \mathbf{u} + \mathbf{f} = \rho \frac{\partial^2 \mathbf{u}}{\partial t^2} \tag{3.74}$$

Okrajové podmínky:

$$\begin{aligned} \mathbf{u} = \mathbf{u}_e \quad &\text{na} \quad \Gamma_e \times (0, T) \\ \mathbf{n} \cdot \boldsymbol{\sigma} = \boldsymbol{\sigma}_n \quad &\text{na} \quad \Gamma_n \times (0, T) \end{aligned}$$

Počáteční podmínky:

$$\begin{aligned} \mathbf{u}(\mathbf{r}, 0) &= \mathbf{u}_0, \quad \mathbf{r} \in \Omega \\ \dot{\mathbf{u}}(\mathbf{r}, 0) &= \dot{\mathbf{u}}_0, \quad \mathbf{r} \in \Omega \end{aligned}$$

Pomocí metody vážených reziduí převedeme rovnici (3.73) na slabou formulaci úlohy. Rovnice pak bude mít tvar:

$$\int_\Omega \rho \mathbf{w} \cdot \ddot{\mathbf{u}} \, \mathrm{d}\Omega + \int_\Omega (\boldsymbol{\beta} \mathbf{w})^T \mathbf{E} \boldsymbol{\beta} \mathbf{u} \, \mathrm{d}\Omega = \int_\Omega \mathbf{w} \cdot \mathbf{f}_V \, \mathrm{d}\Omega + \int_\Gamma \mathbf{w} \cdot \mathbf{f}_B \, \mathrm{d}\Gamma, \tag{3.75}$$

kde $\mathbf{w}$ je testovací funkce, $\mathbf{f}_V$ představuje objemové síly působící uvnitř v oblasti Ω, $\mathbf{f}_B$ představuje plošné síly působící na hranici Γ oblasti Ω.

Podle Galerkinovy metody zvolíme testovací funkci $\mathbf{w}$ a aproximujeme vektor posunutí $\mathbf{u}$. Získáme rovnici:

$$\sum_{a=1}^{n} \sum_{b=1}^{n} \left(\int_\Omega \rho \mathbf{N}_a^T \mathbf{N}_b \, \mathrm{d}\Omega \cdot \widetilde{\ddot{\mathbf{u}}}_b + \int_\Omega (\mathbf{B}_a^u)^T \mathbf{E} \mathbf{B}_b^u \, \mathrm{d}\Omega \cdot \widetilde{\mathbf{u}}_b - \int_\Omega \mathbf{N}_a^T \mathbf{f}_V \, \mathrm{d}\Omega - \int_\Gamma \mathbf{N}_a^T \mathbf{f}_B \, \mathrm{d}\Gamma \right) = 0, \tag{3.76}$$

kde

$$\mathbf{B}_a^u = \boldsymbol{\beta} N_a = \begin{pmatrix} \frac{\partial N_a}{\partial x} & 0 \\ 0 & \frac{\partial N_a}{\partial y} \\ \frac{\partial N_a}{\partial y} & \frac{\partial N_a}{\partial x} \end{pmatrix}. \tag{3.77}$$

Rovnici (3.76) můžeme zapsat v maticovém tvaru:

$$\mathbf{M}_\mathbf{u} \widetilde{\ddot{\mathbf{u}}} + \mathbf{K}_\mathbf{u} \widetilde{\mathbf{u}} = \mathbf{f}, \tag{3.78}$$

kde

$$\mathbf{M}_\mathbf{u} = \coprod_{e=1}^{n_e} \mathbf{m}_\mathbf{u}^e, \quad \mathbf{m}_\mathbf{u}^e = [\mathbf{m}_{pq}], \quad \mathbf{m}_{pq} = \int_{\Omega^e} \rho \mathbf{N}_p^T \mathbf{N}_q \, \mathrm{d}\Omega^e \tag{3.79}$$

$$\mathbf{K}_\mathbf{u} = \coprod_{e=1}^{n_e} \mathbf{k}_\mathbf{u}^e, \quad \mathbf{k}_\mathbf{u}^e = [\mathbf{k}_{pq}], \quad \mathbf{k}_{pq} = \int_{\Omega^e} (\mathbf{B}_p^u)^T \mathbf{E} \mathbf{B}_q^u \, \mathrm{d}\Omega^e \tag{3.80}$$

$$\mathbf{f} = \coprod_{e=1}^{n_e} \mathbf{f}^e, \quad \mathbf{f}^e = [\mathbf{f}_p], \quad \mathbf{f}_p = \int_{\Omega^e} \mathbf{N}_p^T \mathbf{f}_V \, \mathrm{d}\Omega^e + \int_{\Gamma^e} \mathbf{N}_p^T \mathbf{f}_B \, \mathrm{d}\Gamma^e \tag{3.81}$$

Integrál $\int_{\Gamma^e} \mathbf{N}_p^T \mathbf{f}_B \, \mathrm{d}\Gamma^e$ se počítá jen pro hraniční elementy a z matice $\mathbf{N}_p$ se bere její stopa na hranici Γ oblasti Ω. Integrace přes hranici prvku je popsána v části 3.2.1.1

Když použijeme Rayleighův model tlumení, jehož tlumící matici $\mathbf{C_u}$ vypočteme jako lineární kombinaci matice hmoty $\mathbf{M_u}$ a matice tuhosti $\mathbf{K_u}$, tedy

$$\mathbf{C_u} = \alpha_M \mathbf{M_u} + \alpha_K \mathbf{K_u}, \tag{3.82}$$

kde α_M je proporcionální koeficient pro matici $\mathbf{M_u}$ a α_K je proporcionální koeficient pro matici $\mathbf{K_u}$, můžeme rovnici (3.78) přepsat do tvaru:

$$\mathbf{M_u}\widetilde{\ddot{\mathbf{u}}} + \mathbf{C_u}\widetilde{\dot{\mathbf{u}}} + \mathbf{K_u}\widetilde{\mathbf{u}} = \mathbf{f}, \tag{3.83}$$

Na rovnici použijeme Newmarkovu metodu pro získání časové diskretizace. Podle této metody můžeme hyperbolickou diferenciální rovnici pro zvolený časový krok Δt řešit ve třech krocích:

- Prediktorový krok:

$$\begin{aligned}
\widetilde{\mathbf{u}} &= \{\widetilde{\mathbf{u}}\}^n + \Delta t \left\{\widetilde{\dot{\mathbf{u}}}\right\}^n + (1 - 2\beta_H)\frac{\Delta t^2}{2}\left\{\widetilde{\ddot{\mathbf{u}}}\right\}^n, \\
\widetilde{\dot{\mathbf{u}}} &= \left\{\widetilde{\dot{\mathbf{u}}}\right\}^n + \Delta t(1 - \gamma_H)\left\{\widetilde{\ddot{\mathbf{u}}}\right\}^n.
\end{aligned}$$

- Řešení algebraického systému rovnic:

$$\mathbf{M_u^*}\left\{\widetilde{\ddot{\mathbf{u}}}\right\}^{n+1} = \{\mathbf{f}\}^{n+1} - \mathbf{K_u}\widetilde{\mathbf{u}} - \mathbf{C_u}\widetilde{\dot{\mathbf{u}}}$$

  kde

$$\mathbf{M_u^*} = \mathbf{M_u} + \gamma_H \Delta t \mathbf{C_u} + \beta_H \Delta t^2 \mathbf{K_u}$$

- Opravný krok:

$$\begin{aligned}
\{\widetilde{\mathbf{u}}\}^{n+1} &= \widetilde{\mathbf{u}} + \beta_H \Delta t^2 \left\{\widetilde{\ddot{\mathbf{u}}}\right\}^{n+1}, \\
\left\{\widetilde{\dot{\mathbf{u}}}\right\}^{n+1} &= \widetilde{\dot{\mathbf{u}}} + \gamma_H \Delta t \left\{\widetilde{\ddot{\mathbf{u}}}\right\}^{n+1}.
\end{aligned}$$

Ve výše uvedených rovnicích n označuje hodnotu proměnné v čase t a $n+1$ v čase $t + \Delta t$. Časový krok volíme tak, aby byl menší než $2/\omega_{max}$, kde ω_{max} je maximální kruhová frekvence systému.

Pokud zvolíme integrační parametry $\beta_H = 0{,}25$ a $\gamma_H = 0{,}5$, bude se jednat o implicitní metodu. Jestliže bude matice tlumeni $\mathbf{C_u}$ nulová, bude mít tato metoda přesnost druhého řádu.

3.4.3 Programová implementace

Pro vypočtení deformací tělesa důsledkem vlivu vnějších sil je potřeba sestavit matice $\mathbf{M_u}$ a $\mathbf{K_u}$ a vektor sil $\mathbf{f}$ a řešit diferenciální rovnici (3.78). Matice $\mathbf{M_u}$ a $\mathbf{K_u}$ a vektor sil $\mathbf{f}$ sestavíme podle rovnic (3.79), (3.80) a (3.81). K tomu je potřeba vyčíslit uvedené integrály a dílčí matice a vektory poskládat do výsledného tvaru.

Abychom mohli vypočíst uvedené integrály je nutné znát matice $\mathbf{N}$ a $\mathbf{B}$. Posunutí $\mathbf{u}$ v daném bodě je dvourozměrným vektorem, proto použijeme matici bázových funkcí ve tvaru:

$$\mathbf{N}_a = \begin{pmatrix} N_a & 0 \\ 0 & N_a \end{pmatrix} \tag{3.84}$$

Posunutí $\mathbf{u}$ je tedy aproximované nad oblastí elementu Ω^e, který má 6 uzlových bodů, vztahem:

$$\mathbf{u} = \begin{pmatrix} N_1 & 0 & N_2 & 0 & \cdots & N_6 & 0 \\ 0 & N_1 & 0 & N_2 & \cdots & 0 & N_6 \end{pmatrix} \begin{pmatrix} u_{x,1} \\ u_{y,1} \\ u_{x,2} \\ u_{y,2} \\ \vdots \\ u_{x,6} \\ u_{y,6} \end{pmatrix} = \mathbf{N_u}\widetilde{\mathbf{u}}. \tag{3.85}$$

Struktura matice $\mathbf{B}$ je dána rovnicí (3.77). Když tuto matici aproximujeme nad oblastí elementu Ω^e, který má 6 uzlových bodů, bude dána vztahem:

$$\mathbf{B} = \begin{pmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \cdots & \frac{\partial N_6}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & \cdots & 0 & \frac{\partial N6}{\partial y} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \cdots & \frac{\partial N_6}{\partial y} & \frac{\partial N_6}{\partial x} \end{pmatrix}. \tag{3.86}$$

3.5 Mechanika

Pro výpočet rovnovážné pozice tělesa zatíženého vnějšími silami vyjdeme ze stejný vztahů jako v části 3.4. Zajímá nás hodnota posunutí $\mathbf{u}$, při kterém je vnitřní síla rovna vnější síle. Z rovnice (3.73) odstraníme pohybovou složku čímž se nám zjednoduší na tvar:

$$\boldsymbol{\beta}^T \mathbf{E} \boldsymbol{\beta} \mathbf{u} + \mathbf{f} = 0. \tag{3.87}$$

3.5.1 Specifikace podmínek a převod na soustavu lineárních rovnic pomocí MKP

Jsou dány funkce:

$$\begin{aligned} \mathbf{u}_0 &: \Omega \to \Re^2 \\ \dot{\mathbf{u}}_0 &: \Omega \to \Re^2 \\ E, \nu &: \Omega \to \Re \\ \mathbf{f} &: \Omega \to \Re^2 \end{aligned}$$

Hledáme funkci:

$$\mathbf{u} : \Omega_\Gamma \to \Re^2$$

Na celé oblasti platí rovnice (3.87):

$$\boldsymbol{\beta}^T \mathbf{E} \boldsymbol{\beta} \mathbf{u} + \mathbf{f} = 0 \tag{3.88}$$

Okrajové podmínky:

$$\begin{aligned} \mathbf{u} = \mathbf{u}_e \quad &\text{na} \quad \Gamma_e \\ \mathbf{n} \cdot \boldsymbol{\sigma} = \boldsymbol{\sigma}_n \quad &\text{na} \quad \Gamma_n \end{aligned}$$

Pomocí metody vážených reziduí převedeme rovnici (3.87) na slabou formulaci úlohy. Rovnice pak bude mít tvar:

$$\int_\Omega (\boldsymbol{\beta}\mathbf{w})^T \mathbf{E} \boldsymbol{\beta} \mathbf{u} \, \mathrm{d}\Omega = \int_\Omega \mathbf{w} \cdot \mathbf{f}_V \, \mathrm{d}\Omega + \int_\Gamma \mathbf{w} \cdot \mathbf{f}_B \, \mathrm{d}\Gamma, \tag{3.89}$$

kde $\mathbf{w}$ je testovací funkce, $\mathbf{f}_V$ představuje objemové síly působící uvnitř v oblasti Ω, $\mathbf{f}_B$ představuje plošné síly působící na hranici Γ oblasti Ω.

Podle Galerkinovy metody zvolíme testovací funkci $\mathbf{w}$ a aproximujeme vektor posunutí $\mathbf{u}$. Získáme rovnici:

$$\sum_{a=1}^{n} \sum_{b=1}^{n} \left(\int_\Omega (\mathbf{B}_a^u)^T \mathbf{E} \mathbf{B}_b^u \, \mathrm{d}\Omega \cdot \widetilde{\mathbf{u}}_b - \int_\Omega \mathbf{N}_a^T \mathbf{f}_V \, \mathrm{d}\Omega - \int_\Gamma \mathbf{N}_a^T \mathbf{f}_B \, \mathrm{d}\Gamma \right) = 0, \tag{3.90}$$

kde $\mathbf{B}_a^u$ – viz vztah (3.77) Rovnici (3.90) můžeme zapsat v maticovém tvaru:

$$\mathbf{K}_\mathbf{u} \widetilde{\mathbf{u}} = \mathbf{f}, \tag{3.91}$$

kde $\mathbf{K}_\mathbf{u}$ je dáno vztahem (3.80) a $\mathbf{f}$ vztahem (**??**) tj.

$$\mathbf{K}_\mathbf{u} = \coprod_{e=1}^{n_e} \mathbf{k}_\mathbf{u}^e, \quad \mathbf{k}_\mathbf{u}^e = [\mathbf{k}_{pq}], \quad \mathbf{k}_{pq} = \int_{\Omega^e} (\mathbf{B}_p^u)^T \mathbf{E} \mathbf{B}_q^u \, \mathrm{d}\Omega^e \tag{3.92}$$

$$\mathbf{f} = \coprod_{e=1}^{n_e} \mathbf{f}^e, \quad \mathbf{f}^e = [\mathbf{f}_p], \quad \mathbf{f}_p = \int_{\Omega^e} \mathbf{N}_p^T \mathbf{f}_V \, \mathrm{d}\Omega^e + \int_{\Gamma^e} \mathbf{N}_p^T \mathbf{f}_B \, \mathrm{d}\Gamma^e \tag{3.93}$$

Integrál $\int_{\Gamma^e} \mathbf{N}_p^T \mathbf{f}_B \, \mathrm{d}\Gamma^e$ se počítá jen pro hraniční elementy a z matice $\mathbf{N}_p$ se bere její stopa na hranici Γ oblasti Ω.

3.6 Deformace konečnoprvkové sítě volného prostoru

Model aktuátoru se skládá ze dvou částí: lineární pružné těleso a volný prostor. V každém kroku se spočte posunutí mechanických části. Volný prostor není zahrnutý do tohoto výpočtu a tudíž neznáme, jak by se měla zdeformovat síť v oblasti volného prostoru. Podle [5] můžeme z volného prostoru vytvořit pseudo-elastické těleso tak, že nastavíme pro každý element sítě (v našem případě šestiuzlový trojúhelník) virtuální modul pružnosti a v samostatném

výpočtu určíme posunutí **u** pro uzly z oblasti volného prostoru. Virtuální Youngův modul pružnosti je dán vztahem:

$$E = \frac{1}{\sqrt{d\mu(T)}}, \tag{3.94}$$

kde d je vzdálenost středu elementu k nejbližší zdi a $\mu(T)$ je plocha elementu.

Posunutí mechanické části se použije jako okrajová podmínka. Výsledkem bude posunutí uzlů volného prostoru podle pohybu mechanické části.

3.7 Výpočet elektrostatické síly na povrchu elektrody

Vycházíme z Gaussova zákona elektrostatiky

$$Q = \varepsilon \oint_S \mathbf{E}\,\mathrm{d}\mathbf{S}, \tag{3.95}$$

kde Q je elektrický náboj uvnitř uzavřené plochy S, ε je permitivita prostředí a $\mathbf{E}$ je intenzita elektrostatického pole, a ze vztahu pro intenzitu elektrostatického pole

$$\mathbf{F} = Q\mathbf{E}, \tag{3.96}$$

kde $\mathbf{F}$ je síla působící na bodový náboj Q v místě s intenzitu elektrostatického pole $\mathbf{E}$.

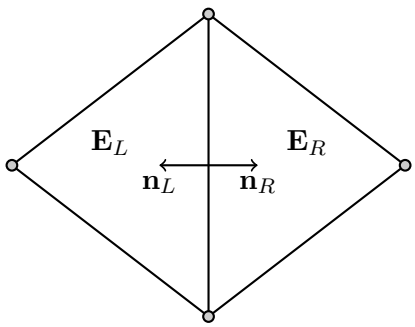
Tyto rovnice lze upravit pro případ síly působící na hranici nespojitosti dvou elektrostatických polí jako

$$\mathbf{F} = Q\mathbf{E}_R + Q\mathbf{E}_L, \tag{3.97}$$

kde

$$Q = l(\varepsilon_L \mathbf{E}_L \cdot \mathbf{n}_L + \varepsilon_R \mathbf{E}_R \cdot \mathbf{n}_R) \tag{3.98}$$

je náboj na hranici nespojitosti, l je délka hranice a $\mathbf{n}$ je jednotková normála. Schématicky jsou jednotlivé veličiny znázorněny na obrázku 3.3.



Obrázek 3.3: Značení pro výpočet intenzity na hraně

V oblasti elektrody je intenzita elektrického pole nulová, takže rovnice (3.97) a (3.98) můžeme upravit tak, že v nich zůstane jen jeden člen.

Kapitola 4

Realizace

Program je realizován v jazyce Java. Jako vývojové prostředí bylo použito programu IntelliJ IDEA. K řešení úlohy je implementovaná metoda konečných prvků a pro časovou diskretizaci je použita Newmarkova metoda, která byla popsána v předchozí kapitole. Pro řešení soustavy lineárních rovnic se používá metoda konjugovaných gradientů s předpodmíněním pomocí nekompletní Cholesky dekompozice.

Program se skládá z následujících komponent: generátor sítě jDistMesh, parser vstupních dat a jejich reprezentace, řešení elektrostatického modelu, řešení pro statický mechanický model, řešení pro dynamický mechanický model, propojení elektrostatické části s dynamickou a mechanickou částí, generátor výstupu. Bližší popsání těchto komponent se nachází v této kapitole.

4.1 Formát vstupních souborů

Program pro svůj výpočet potřebuje nějaké zadání. Toto zadání musí obsahovat popis oblasti, okrajové a počáteční podmínky, vnější působení sil, materiálové vlastnosti. Oblast musí být rozdělena na trojúhelníky. Protože program potřebuje pro svůj výpočet kvadratické trojúhelníky, musí být každý trojúhelník určen 6 uzly.

Vstupní formát vychází z formátu, který popisuje Jonathan Richard Shewchuk pro svůj program triangle [16]. Vstupní data jsou uložena ve třech souborech:

- `<filename>.node`
- `<filename>.ele`
- `material.mat`

Všechny tři soubory jsou textové, jejich syntaxe je popsána níže.

4.1.1 .node

Tento soubor popisuje umístění jednotlivých uzlů. Jako komentář se používá znak #. Vše, co je uvedeno za tímto znakem až do konce řádku, je ignorováno. Jednotlivé hodnoty jsou odděleny alespoň jedním bílým znakem.

První řádek obsahuje hlavičku souboru. Má následující strukturu:

```
<#nodes> <dimension> <#attributes> <#boundary_markers>
```

#nodes *int*, počet uzlů.

dimension *int*, počet souřadnic. Vždy 2.

#attributes *int*, počet atributů. Povolené jsou pouze následující hodnoty: 1, 3, 5, 7.

#boundary_markers *int*, počet příznaků. Vždy 1.

Následující řádky obsahují informace o jednotlivých uzlech. Mají následující formát:

```
<node #> <x> <y> [attributes] <boundary_marker>
```

node # *int*, index uzlu. Uzly jsou indexovány od jedničky.

x *double* x-ová souřadnice. Jednotka $[m]$

y *double* y-ová souřadnice. Jednotka $[m]$

attributes hodnoty atributu v tomto uzlu

1. hodnota elektrostatického potenciálu. Jednotka $[V]$
2. x-ová hodnota posunutí. Jednotka $[m]$
3. y-ová hodnota posunutí. Jednotka $[m]$
4. x-ová složka plošné síly. Jednotka $[N \cdot m^{-2}]$
5. y-ová složka plošné síly. Jednotka $[N \cdot m^{-2}]$
6. x-ová složka objemové síly. Jednotka $[N \cdot m^{-3}]$
7. y-ová složka objemové síly. Jednotka $[N \cdot m^{-3}]$

boundary_marker *int* bitové příznaky. Toto celé číslo v sobě může zahrnovat několik bitových příznaků. Jedná se o bitový součet jednotlivých příznaků.

2 (0x02) pokud je 2. bit nastaven na jedničku, použije se 2. a 3. atribut jako zadané posunutí tohoto uzlu. Tyto atributy se použijí jako okrajová podmínka pro výpočet mechaniky. Pokud je bit nastaven na nulu jsou 2. a 3. atribut ignorovány.

4 (0x04) pokud je 3. bit nastaven na jedničku, použije se 1. atribut jako hodnota elektrostatického potenciálu v tomto uzlu. Tento atribut se použije jako okrajová podmínka pro výpočet elektrostatiky. Pokud je bit nastaven na nulu je první atribut ignorován.

8 (0x08) pokud je 4. bit nastaven na jedničku, použije se 6. a 7. atribut jako zadaná objemová síla, která působí v tomto bodě. Pokud je bit nastaven na nulu jsou tyto atributy ignorovány.

16 (0x10) pokud je 5. bit nastaven na jedničku, je požadováno vytisknutí souřadnic tohoto uzlu na standardní výstup.

32 (0x20) pokud je 6. bit nastaven na jedničku, použije se 4. a 5. atribut jako zadaná plošná síla, která působí v tomto bodě. Pokud je bit nastaven na nulu jsou tyto atributy ignorovány.

64 (0x40) pokud je 7. bit nastaven na jedničku, bude uzel považován za zeď. Tento příznak se použije pouze pro transformaci prostředí mezi elektrodami při výpočtu aktuátoru.

Ukázka tohoto souboru je uvedena v příloze B.1 na straně 57.

4.1.2 .ele

Tento soubor popisuje z jakých uzlů jsou složeny jednotlivé trojúhelníky. Každý trojúhelník je určen šesti uzly. Jako komentář se používá znak #. Vše, co je uvedeno za tímto znakem až do konce řádku, je ignorováno. Jednotlivé hodnoty jsou odděleny alespoň jedním bílým znakem.

První řádek obsahuje hlavičku souboru. Má následující strukturu:

```
<#elements> <#nodes> <#attributes>
```

#elements *int*, počet elementů – trojúhelníků.

#nodes *int*, počet uzlů, které určují jeden trojúhelník, tedy vždy 6.

#attributes *int*, počet atributů. Vždy 1, pouze ID materiálu, viz níže.

Následující řádky obsahují informace o jednotlivých trojúhelnících. Mají následující formát:

```
<element #> <node1> <node2> <node3> <node4> <node5> <node6> <ID materialu>
```

element # *int*, index trojúhelníku. Trojúhelníky jsou indexovány od jedničky.

node1 *int*, index 1. uzlu. Uzel leží ve vrcholu trojúhelníku.

node2 *int*, index 2. uzlu proti směru hodinových ručiček. Uzel leží ve vrcholu trojúhelníku.

node3 *int*, index 3. uzlu proti směru hodinových ručiček. Uzel leží ve vrcholu trojúhelníku.

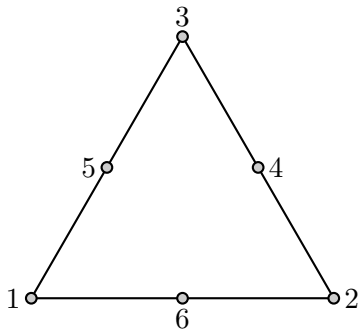
node4 *int*, index 4. uzlu. Uzel leží na hraně trojúhelníku, která je protilehlá 1. uzlu.

node5 *int*, index 5. uzlu. Uzel leží na hraně trojúhelníku, která je protilehlá 2. uzlu.

node6 *int*, index 6. uzlu. Uzel leží na hraně trojúhelníku, která je protilehlá 3. uzlu.

ID materialu *int*, identifikátor materiálu. Index do seznamu materiálu, viz 4.1.3.

Jak jsou číslovány uzly v rámci jednoho trojúhelníku, je znázorněno na obrázku 4.1. Ukázka tohoto souboru je uvedena v příloze B.2 na straně 57.



Obrázek 4.1: Jak jsou číslovány uzly ve vstupním souboru v rámci jednoho trojúhelníku

4.1.3 .mat

Tento soubor je materiálovým číselníkem. Každý materiál má svůj identifikátor a sadu vlastností. Jako komentář se používá znak #. Vše, co je uvedeno za tímto znakem až do konce řádku, je ignorováno. Jednotlivé hodnoty jsou odděleny alespoň jedním bílým znakem.

První řádek obsahuje hlavičku souboru. Má následující strukturu:

```
<#materials> <#attributes>
```

#materials *int*, počet materiálů v číselníku.

#attributes *int*, počet atributů. Vždy 6.

Následující řádky obsahují informace o jednotlivých materiálech. Mají následující formát:

```
<materialId> <rho> <E> <nu> <permitivita> <thickness> <isMechanicArea>
```

materialId *int*, identifikátor materiálu. Materiál může mít libovolný celočíselný identifikátor. Jednotlivé identifikátory netvoří posloupnost.

rho *double*, hustota. Jednotka $[kg \cdot m^{-3}]$

E *double*, Youngův modul pružnosti v tahu. Jednotka $[Pa]$

nu *double*, Poissonovo číslo. Jednotka [1]

permitivita *double*, permitivita. Jednotka $[F \cdot m^{-1}]$

thickenss *double*, tloušťka materiálu. Měla by být v celém zadání konstantní. Jednotka $[m]$

isMechanicArea *int*, příznak, který určuje, jestli se bude oblast s tímto materiálem během výpočtu aktuátoru počítat pomocí mechaniky (pokud je 1) nebo pomocí elektrostatiky (pokud je 0).

Ukázka tohoto souboru je uvedena v příloze B.3 na straně 57.

4.2 jDistMesh

Původním záměrem bylo pro generování sítě používat program Triangle [16], ale ten se ukázal v průběhu vývoje jako nevyhovující. Proto byl vytvořen program jDistMesh.

Program vznikl přepsáním knihovny DistMesh [14] (ta je naprogramovaná v Matlabu) do jazyka Java. Knihovna není přepsána celá, ale jen část knihovny potřebná pro generování 2D sítí. Vedle generování 2D sítě program umožňuje zápis vygenerované sítě do souboru ve formátu, který je popsán v části 4.1. Program poskytuje API pro integraci do jiného programu. Generování sítě je možné i bez integrování do jiného programu a to tak, že vytvoříme vstupní soubor v jazyce JavaScript, na základě kterého program jDistMesh vygeneruje požadovanou síť.

Původní knihovna používá vestavěnou funkci `delaunayn`[19], která nad množinou uzlů vytvoří Delaunay triangulaci. Delaunay triangulace je taková triangulace, kde pro každý trojúhelník platí: kružnice opsaná trojúhelníku obsahuje z celé množiny uzlů jen vrcholy daného trojúhelníku. Na vyřešení této úlohy je použita část zdrojových kódů z programu "Voronoi Diagram / Delaunay Triangulation"[4].

Program byl vyvinut a testován na běhovém prostředí Java 6.

4.2.1 Seznam komponent a popis rozhraní

Každá funkce z knihovny DistMesh byla převedena na jednu třídu. Třídy jsou rozděleny do dvou balíčků: `cz.cvut.fel.plichjan.distmesh` a `cz.cvut.fel.plichjan.distmesh.inputs`.

4.2.1.1 Třídy definující vlastnosti sítě z `distmesh.inputs`

Jeden ze vstupních parametrů generátoru sítě je funkce (interface) definující vzdálenost od hranice oblasti. Třídy definující tvar oblasti implementují rozhraní `IDistanceFunction`. Toto rozhraní předepisuje jedinou funkci `double call(double x, double y)`, která vrátí znaménkovou vzdálenost od hranice oblasti, tj. pokud bod `[x,y]` leží uvnitř oblasti, bude vrácená hodnota záporná, pokud leží vně, oblasti bude hodnota kladná.

DCircle(x, y, r) Oblast ve tvaru kružnice s poloměrem `r` a středem v bodě `[x,y]`.

DDiff(set1, set2) Oblast, která vznikne, když od oblasti `set1` odebereme vnitřní body oblasti `set2`.

DIntersect(set1, set2) Oblast, která vznikne jako průnik oblasti `set1` a oblasti `set2`.

DPoly([x, y], ...) Oblast ve tvaru mnohoúhelníku.

DRectangle(x1, x2, y1, y2) Oblast ve tvaru obdélníku s vrcholy `[x1,y1]`, `[x2,y1]`, `[x1,y2]`, `[x2,y2]`. Rychlá, ale nepřesná implementace. Nevhodné pro skládání s dalšími tvary.

DRectangle0(x1, x2, y1, y2) Oblast ve tvaru obdélníku s vrcholy `[x1,y1]`, `[x2,y1]`, `[x1,y2]`, `[x2,y2]`. Přesná implementace.

DUnion(set1, set2) Oblast, která vznikne jako sjednocení oblasti `set1` a oblasti `set2`.

Třídy definující velikost hran sítě implementují rozhraní `IEdgeLengthFunction`. Toto rozhraní předepisuje jedinou funkci `double call(double x, double y)`, která vrátí požadovanou velikost hrany v bodě o souřadnicích `[x,y]`.

HUniform() Třída určující v celé oblasti stejnou velikost hran sítě.

4.2.1.2 Třídy pracující se sítí z `distmash`

BndProj() Tato třída přesune uzly, které jsou na okraji sítě tak, aby ležely na okraji zadané oblasti.

BoundEdges() Tato třída nalezne hrany, které jsou na okraji sítě, tj. hrany, které náleží jen jednomu trojúhelníku.

DistMesh2D() Tato třída vygeneruje síť podle zadaných parametrů.

FixMesh() Tato třída – na rozdíl od funkce v Matlabu – pouze nastaví všem trojúhelníkům stejnou orientaci a to proti směru hodinových ručiček.

ViewFrame Tato třída slouží k zobrazování sítě v průběhu generování.

SimpVol() Tato třída spočte znaménkový objem resp. obsah zadaného simplexu. Používá se ke zjištění orientace trojúhelníku.

4.2.1.3 Přidané třídy

Tyto třídy nemají vzor v žádné funkci z knihovny DistMesh. Některé z nich vnikly na základě požadavků na vstupní data pro program jFEM, který je popsán v části 4.3. Byly přidány např. třídy pro ukládání a načítání sítě a jedna třída pro doplnění uzlů uprostřed hran.

AddMidpoints() Tato třída přidá uzly, které leží uprostřed hran. Znamená to, že se přidá tolik uzlů, kolik je hran v dané síti, a z trojuzlových trojúhelníků se stanou šestiuzlové.

BoundNodes() Tato třída najde všechny uzly, které leží na okraji sítě.

NodePrinter(filename) Tato třída poskytuje rozhraní pro zapsání souřadnic uzlů do souboru ve formátu `.node`, viz část 4.1.1 na straně 23.

ElePrinter(filename) Tato třída poskytuje rozhraní pro zapsání trojúhelníků do souboru ve formátu `.ele`, viz část 4.1.2 na straně 25.

MeshReader(filename) Tato třída poskytuje rozhraní pro načtení sítě. Načtení se provede ze dvou souborů: `filename.node` a `filename.ele`.

FreeFemMeshReader(filename) Tato třída poskytuje rozhraní pro načtení sítě ze souboru `filename.msh`, který je vygenerovaný programem *FreeFem++*. Podrobný popis formátu <http://www.ann.jussieu.fr/~lehyaric/ffcs/doc/ff/ffdocsu17.html>

4.2.1.4 Pomocné funkce v JavaScriptu

Při návrhu sítí pro program *jFEM* byla vytvořena sadu pomocných funkcí, konstant a struktur v JavaScriptu, které zjednodušují práci při ukládání vygenerované sítě do souboru. Umožňují snadno zadat uzlové uzlové veličiny a zvolit typ materiálu. Kompletní přehled je v příloze C.

V části 5.1.1 je ukázka skriptu 5.1, který vygeneruje síť zobrazenou na obrázku 5.2. Červenými kroužky jsou označeny uzly uprostřed hran.

4.3 jFEM

Pro sestavování (buildu) projektu je použit program Apache Maven 3.0 [18]. Pomocí tohoto programu byla vytvořena i základní adresářová struktura projektu. Projekt využívá následujících 5 knihoven:

Apache Log4j 1.2 [17] Tato knihovna se využívá k logování událostí v průběhu výpočtu.

Guava 16.0 [8] Tato knihovna se využívá při testování a při práci s kolekcemi objektů.

MTJ 1.0 [9] Matrix Toolkits for Java – jak zní nezkrácený název této knihovny – poskytuje sadu tříd na reprezentaci řídkých i hustých matic. Obsahuje i balíček tříd – implementací algoritmů – na řešení řídké soustavy lineárních rovnic (např. CG, GMRES). V programu *jFEM* je pro reprezentaci řídké matice zvolena třída `CompRowMatrix`. Při použití této třídy je potřeba dopředu vědět, v kterých místech by mohli být nenulové prvky matice. To lze zjistit ze struktury sítě. Pokud uzly s indexy i a j jsou součásti nějakého trojúhelníku, tak prvek matice m_{ij} může být nenulový. Toto platí, pokud matici násobím skalární veličinou. Pokud by se jednalo o vektorovou veličinu dimenze d, tak mohou být nenulové všechny prvky matice m_{kl}, kde $k = d \cdot i + a$ a $l = d \cdot j + b$ pro všechny $a, b \in \{0, 1, \ldots, d-1\}$[1]. Na této knihovně je postavena velká část této aplikace.

netlib-java [10] Tato knihovna je využívána knihovnou MTJ. Je to knihovna funkcí pro lineární algebru. Knihovna je napsaná jako wrapper knihoven BLAS, LAPACK a ARPACK. Vedle nativní podpory pro hlavní operační systémy knihovna obsahuje i čistě Javovou implementaci těchto knihoven vzniklou pomocí F2J pro zajištění přenositelnosti.

JUnit 4 [1] Tato knihovna se využívá při psaní jednotkových testů.

Projekt je rozdělen do několika komponent. Jsou tu komponenty, které reprezentují isoparametrické elementy 2D a 1D. Projekt obsahuje také sadu tříd, které řídí sestavování matic a vektorů levých stran a i průběh výpočtu řešení a sadu pomocných `"Tools"` tříd. Tyto třídy mají jen statické metody a jsou vlastně knihovnou funkcí. Do této sady tříd patří `ParamTools`, která zjednodušuje načítání parametrů z příkazové řádky, `PrintTools`, která má sadu metod na výpis hodnot na standardní výstup a `Tools`, která má mnoho metod, které zjednodušují práci při sestavování matic.

Pro účely předávání informací v průběhu řešení soustavy byla vytvořena dvě rozhraní:

[1] V tomto případě jsou indexy uzlů i prvků matice jsou počítány od nuly

`IDynamicFEMLoopListener`,
: které se používá při řešení dynamických problémů, při nichž je soustava vyhodnocována v jednotlivých časových krocích
Toto rozhraní implementují např. třídy `ViewListener`, `PrintAllNodesListener` a `ListLoopListener`.

`IStaticFEMResultListener`,
: které se používá pro řešení statických problémů.
Toto rozhraní implementují např. třídy `ViewListener`, `ListResultListener` a `PrintAllNodesListener`

Implementace těchto rozhraní se používají např. pro výpis výsledků do souboru či na standardní výstup, nebo pro vizualizaci.

4.3.1 QuadraticIsoparam

Tato třída reprezentuje šestiuzlový isoparametrický trojúhelník. Umožňuje spočítat všechny potřebné integrály nad tímto elementem, tj. (3.60), (3.79), (3.80) a první integrál ze vzorce (3.81). Také poskytuje universální funkci, pomocí níž se dá spočítat libovolný integrál nad tímto elementem. K tomuto účelu disponuje funkcemi na výpočet hodnoty tvarových funkcí pro skalární i vektorové 2D funkce v zadaných ξ_i, derivací podle všech ξ_i (3.21), derivací podle x a y (3.43), jakobiánu (det $\mathbf{J}$) (3.41) a matice $\mathbf{B}$ (3.86). Dále nabízí funkce na sestavení matice $\mathbf{E}$ (3.66) podle Laméových koeficientu λ_L a μ_L nebo podle Youngova modulu pružnosti E a Poissonovy konstanty ν. V neposlední řadě umožňuje z rozložení elektrostatického potenciálu nad tímto elementem spočítat plošnou sílu $\mathbf{f}_B$ (tj. sílu na jednotku plochy) (3.97), která působí na danou stranu trojúhelníku.

```
144 public <V> V integrateElem(final double[] x, final double[] y, V sum,
        IFunction<V> function, ISummation<V> summation) {
145   assert x.length == y.length;
146   for (int i = 0; i < weightsS.length; i++) {
147     V v = function.call(xiS[i], x, y);
148     sum = summation.add(sum, weightsS[i] * jacobian(xiS[i], x, y), v);
149   }
150   return sum;
151 }
```

Ukázka kódu 4.1: Universální funkce pro výpočet integrálu

```
207 public Matrix integrateElemM2D(final double ro, final double[] x, final
        double[] y) {
208   int dim = 2 * x.length;
209   return integrateElem(x, y, new DenseMatrix(dim, dim),
210     new IFunction<Matrix>() {
211       @Override
212       public Matrix call(double[] xi, double[] x, double[] y) {
213         return new DenseMatrix(fceND2(xi));
214       }
215     },
216     new ISummation<Matrix>() {
217       @Override
```

```
218         public Matrix add(Matrix sum, double alpha, Matrix n) {
219           return sum.transRank1(ro * alpha, n);
220         }
221       }
222     );
223   }
```

Ukázka kódu 4.2: Výpočet integrálu podle vzorce (3.79)

4.3.2 QuadraticIsoparamLine

Tato třída reprezentuje trojuzlový jednorozměrný element. Umožňuje spočítat druhý integrál ze vzorce (3.81). Výpočet provádí podobně jako třída `QuadraticIsoparam` jen v jednom rozměru a tvarové funkce mají pouze dva parametry ξ_1 a ξ_2.

4.3.3 ElectrostaticFEM

Tato třída implementuje postup pro výpočet rozložení elektrostatického potenciálu tak, jak je popsán v části 3.3. Při výpočtu aktuátoru se používá ve chvíli, kdy je potřeba zjistit rozložení elektrostatického potenciálu. Postup výpočtu se dá shrnout do následujících kroků: příprava struktury matice soustavy podle zadané sítě, spočtení prvků matice a vektoru pravých stran projitím přes všechny elementy sítě, aplikace okrajové podmínky na soustavu, vypočtení řešení soustavy.

Pokud by uživatel programu *jFEM* chtěl zjistit pouze rozložení elektrostatického potenciálu, použije tuto třídu samostatně.

4.3.4 MechanicsFEM

Tato třída implementuje postup pro výpočet posunutí lineárního pružného tělesa tak, jak je popsán v části 3.5. Při výpočtu aktuátoru se používá k přípravě sítě pro výpočet elektrostatického potenciálu. Je to v okamžiku, kdy je potřeba přesunout uzly sítě v okolí mechanických částí podle toho, jak se posunuly mechanické části aktuátoru.

Vhodným vylepšením do budoucna by byla výkonnostní optimalizace této třídy, která by umožnila spočítat matici tuhosti dopředu pro opakované použití s jinými okrajovými podmínkami. Během výpočtu aktuátoru by se pouze nastavily okrajové podmínky (tj. posunutí krajních uzlů sítě) a pak by se celá soustava vyřešila. Tato optimalizace by urychlila výpočet aktuátoru.

Tato třída se dá použít i samostatně k výpočtu rovnovážného stavu mechanického sytému zatíženého vnějšími silami, které se nemění při posunutí.

4.3.5 DynamicFEM

Tato třída je jedna ze dvou hlavních komponent systému. Implementuje Newmarkovu metodu na výpočet chování mechanického lineárního systému v jednotlivých časových krocích

popsanou v části 3.4. Na začátku každého kroku je zavolán `IDynamicFEMLoopListener`, který může změnit např. hodnoty vnějších sil, které působí na soustavu.

Při výpočtu aktuátoru se používá k zjištění posunutí mechanických částí v jednotlivých časových okamžicích. Dá se použít i samostatně pro výpočet dynamického chování mechanického systému zatíženého konstantními vnějšími silami.

4.3.6 ActuatorFEM

Toto je hlavní třída, která zajišťuje výpočet aktuátoru. Ona sama žádnou soustavu rovnic neřeší. Jejím úkolem je rozdělení vstupního zadání na mechanickou a elektrostatickou část, vytvoření dílčího zadání pro podřízené komponenty `DynamicFEM`, `ElectrostaticFEM` a `MechanicsFEM` a přepočet rozložení elektrostatického potenciálu na plošné síly působící na mechanickou část. Pro propojení `DynamicFEM` s ostatními komponentami vytváří anonymní `IDynamicFEMLoopListener`, který použije posunutí uzlů z mechanické oblasti k úpravě sítě pro elektrostatickou oblast. Následně je vypočítáno rozložení potenciálu a z něj je vypočítána síla, která působí na elektrody tj. na mechanickou oblast.

Třída `ActuatorFEM` načte zadání úlohy tj. vstupní síť s parametry a vytvoří tři nové sítě. Každá síť obsahuje všechny uzly z původní sítě. Trojúhelníky do nových sítí jsou přidány jen tehdy, pokud jsou v dané síti potřeba. Podrobnější vysvětlení je v odstavcích níže. Protože jsou zahrnuty všechny uzly, není potřeba provádět přeindexování trojúhelníků a i převod sil z elektrostatické oblasti do mechanické je jednodušší.

Bylo potřeba vymyslet, jak zahrnout do soustavy uzly, které nejsou použity v trojúhelnících. Řešením bylo přidat rovnice typu $x_i = 0$. Tím se do soustavy rovnic zahrnuly všechny uzly z původní sítě.

Pro `DynamicFEM` je síť tvořena pouze trojúhelníky, které patří do mechanické oblasti. Data pro trojúhelníky jsou převzata ze zadané sítě. A uzlová data jsou zkopírována jen pro uzly z mechanické oblasti. Zadané síly jsou nahrazeny nulovými.

Pro `MechanicsFEM` je síť tvořena pouze trojúhelníky z nemechanické oblasti (tj. volný prostor). Pro tyto trojúhelníky je spočítán virtuální Youngův modul pružnosti podle rovnice (3.94). K tomu je potřeba zjistit vzdálenost od nepohyblivých částí. K tomu účelu slouží atribut každého uzlu sítě, který říká, že je jedná o zeď. Tato operace se provede v inicializační části před iterační smyčkou. A uzlová data jsou zkopírována jen pro uzly z nemechanické oblasti. Zadané síly jsou nahrazeny nulovými.

Pro `ElectrostaticFEM` je síť tvořena pouze trojúhelníky z nemechanické oblasti (tj. volný prostor). Data pro trojúhelníky jsou převzata ze zadané sítě. A uzlová data jsou zkopírována jen pro uzly z nemechanické oblasti. Zadané síly jsou nahrazeny nulovými. Vzhledem k tomu, že pozice uzlů bude spočítána v každém kroku znovu, je pro ně vytvořena pouze kolekce.

4.3.7 Reprezentace sítě

Pro reprezentaci zadání úlohy slouží třída `Mesh`. Jedná se de-facto o objektovou reprezentaci vstupního souboru. Atribut `nodes` obsahuje seznam souřadnic jednotlivých uzlů. Atribut `triangles` obsahuje seznam informací o jednotlivých trojúhelnících. Tyto informace jsou

reprezentovány třídou `TriangleData`. Jedná se o pole indexu uzlů, z kterých se trojúhelník skládá, a materiálové vlastnosti: hustota, permitivita, Youngův modul pružnosti, Poissonuv koeficient, tloušťka materiálu a příznak, zda je daný trojúhelník součástí mechanické oblasti.

Další atribut `nodeDataList` třídy `Mesh` obsahuje seznam informací, které byli zadány k jednotlivým uzlům. Informace jsou zadány pouze u některých uzlů, proto nejsou tyto informace připojeny přímo k souřadnicím uzlu tak, jak je tomu u trojúhelníků. Proto třída `NodeData`, která tyto informace reprezentuje, obsahuje index uzlu, ke kterému tato informace patří. Dalšími zadanými atributy jsou: elektrický potenciál, vektor posunutí a příznak, že daný uzel je zeď. Také je zde příznak, zda se má uzel tisknout.

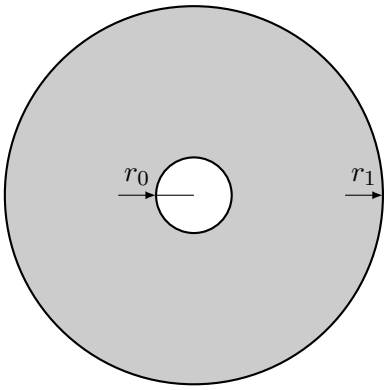
Další atribut `forces` třídy `Mesh` obsahuje seznam objemových sil pro všechny uzly sítě. Atribut `boundaryForces` obsahuje seznam plošných sil Ten se používá zároveň s atributem `boundary`, který obsahuje seznam hran na hranici oblasti. Každá hrana má atribut `trianglge` s daty o trojúhelníku, ke kterému patří, atribut `ix` se seznamem indexu uzlů a atribut `i`, což je index hrany v rámci trojúhelníku, ke kterému patří.

Kapitola 5

Testování

5.1 ElektrostaticFEM

5.1.1 Testování proti analytickému řešení



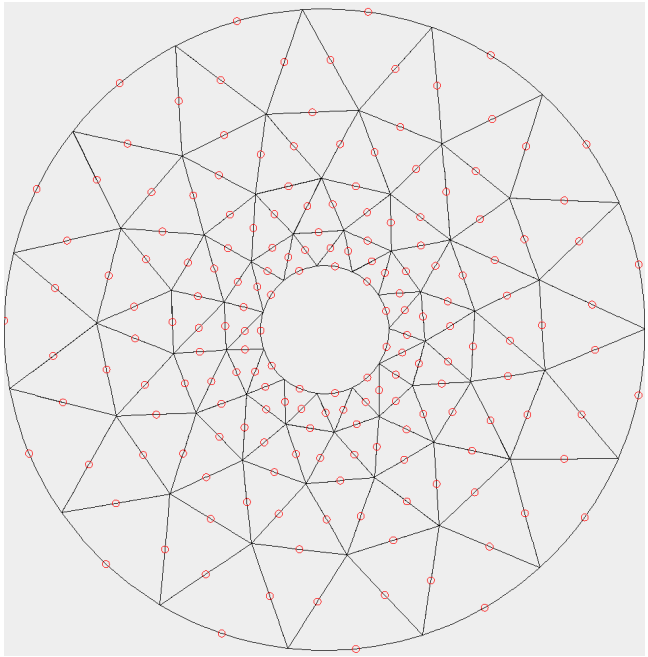
Obrázek 5.1: Schéma zadání pro výpočet rozložení potenciálu

Pro otestování výpočtu rozložení potenciálu byla zvolena oblast ve tvaru mezikruží zobrazená na obrázku 5.1. Hodnota potenciálu – okrajová podmínka – je zadána na hranicích oblasti. Na vnitřním kruhu je hodnota potenciálu rovna jedné a na vnějším kruhu je hodnota potenciálu rovna nule. Budeme hledat rozložení potenciálu mezi oběma kruhy.

Na vygenerování zadání pro program *jFEM* použijeme programu *jDistMesh*. Vstupem mu bude skript 5.1, kterým vygenerujeme síť zobrazenou na obrázku 5.2.

```
var r0 = 1, r1 = 5;
var v0 = 1, v1 = 0;
var c0 = new DCircle(0, 0, r0);
var c1 = new DCircle(0, 0, r1);
var h0 = 0.5;
distMesh2D.dptol = 0.0001;
var df = new DDiff(c1, c0);

var mesh2 =
```



Obrázek 5.2: Síť pro testování výpočtu rozložení elektrostatického potenciálu

```
distMesh2D.call(
        df,
        function call(x, y) {
            return h0+0.3*c0.call(x,y);
        },
        h0, [[-r1, -r1], [r1, r1]],
        []
);

mesh2 = new AddMidpoints().call(mesh2.p, mesh2.t);
mesh2.p = new BndProj().call(mesh2.p, mesh2.t, df);
viewFrame.drawMesh(mesh2);

var file = "circleNu01";
storeMesh(file, mesh2, h0 / 4, [
    {df: c0, flags: IS_SET_POTENTIAL, potential: v0 },
    {df: c1, flags: IS_SET_POTENTIAL, potential: v1 }
], [
    {df: forEvery, matId: 10}
]);
```

Ukázka kódu 5.1: Skript na vygenerování sítě pro testování `ElektrostaticFEM`

5.1.1.1 Výpočet analytického řešení

Vyjádříme si Laplaceovu rovnici v polárních souřadnicích.

$$\Delta u = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 u}{\partial \varphi^2} = 0$$

Protože je úloha středově souměrná, závisí hodnota u jen na r, tedy na vzdálenosti od středu souměrnosti. Proto se rovnice pro tuto úlohu zjednoduší na tvar

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) = 0$$

Rovnici vynásobíme r a po integraci a následujících úpravách získáme

$$\begin{aligned} r\frac{\partial u}{\partial r} &= k_1, \\ \frac{\partial u}{\partial r} &= \frac{k_1}{r}, \\ u(r) &= k_1 \ln r + k_2 \end{aligned}$$

Konstanty k_1 a k_2 určíme z okrajových podmínek

$$\begin{aligned} u(r_0) &= U_0, \\ u(r_1) &= U_1. \end{aligned}$$

Řešením této okrajové úlohy je výraz

$$u(r) = (U_0 - U_1)\frac{\ln(r) - \ln(r_1)}{\ln(r_0) - \ln(r_1)} + U_1.$$

Když do této rovnice dosadíme konkrétní hodnoty

$$\begin{aligned} r_0 = 1, \quad U_0 = 1, \\ r_1 = 5, \quad U_1 = 0 \end{aligned}$$

tak řešením bude výraz

$$u(r) = \frac{\ln(r) - \ln(5)}{\ln(1) - \ln(5)}.$$

5.1.1.2 Porovnání výsledků

Necháme program vypočíst řešení této úlohy pro různě husté sítě. Sítě se boudou postupně zjemňovat a budou mít stále více trojúhelníků. Chybu budeme sledovat na celé oblasti Ω. L_2 norma chyby je aproximována jako

$$\|err\| = \sqrt{\int_\Omega (u - \tilde{u})^2 \,\mathrm{d}\Omega} = \sqrt{\sum_i^{n_e} \int_{\Omega_e} (u - \tilde{u})^2 \,\mathrm{d}\Omega_e},$$

h	$\|\|err\|\|$
0.109846	1.372936998
0.0305118	0.680122048
0.0139806	0.461508396
0.00799936	0.339022713
0.0051792	0.26809737
0.00362279	0.219494419
0.00267361	0.192296646
0.00206595	0.163052752
0.00164349	0.149135509
0.00136071	0.131307882
0.00115322	0.12215351
0.000951411	0.112343491
0.000856096	0.103873384

Tabulka 5.1: Závislost chyby numerického řešení na průměrné délce strany trojúhelníku.

kde integrál v druhé části výrazu je aproximován tak, jako ostatní integrály nad elementem pomocí Gaussových bodů tj.

$$\int_{\Omega_e} (u - \tilde{u})^2 \, \mathrm{d}\Omega_e \approx \sum_{i}^{n_g} w_i (\mathbf{N}(\xi_1^i, \xi_2^i, \xi_3^i) \cdot \widetilde{\mathbf{err}}_e)^2 \det(\mathbf{J})$$

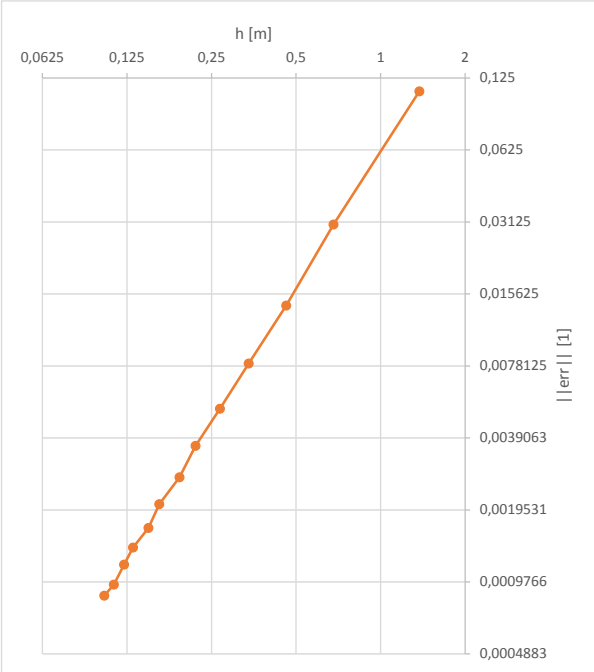
kde $\widetilde{\mathbf{err}}_e$ je vektor chyb v uzlových bodech elementu e. Výsledky jsou shrnuty do tabulky 5.1 a zobrazeny v grafu na obrázku 5.3. Graf má logaritmické obě osy. Hodnota h je průměrná délka strany trojúhelníku.

Z grafu je vidět, že když síť zjemníme tak, že průměrná délka strany trojúhelníku bude poloviční, zmenší se chyba na čtvrtinu. Tímto je otestováno, že numerické řešení konverguje k analytickému, neboť chyba řešení klesá se zjemňováním sítě.

5.2 MechanicFEM

5.2.1 Matice hmotnosti

V sadě jednotkových testů pro třídu `QuadraticIsoparam` je také test výpočtu matice hmotnosti podle vzorce 3.79. Výpočet se provede pro trojúhelník zobrazený na obrázku 5.4 a



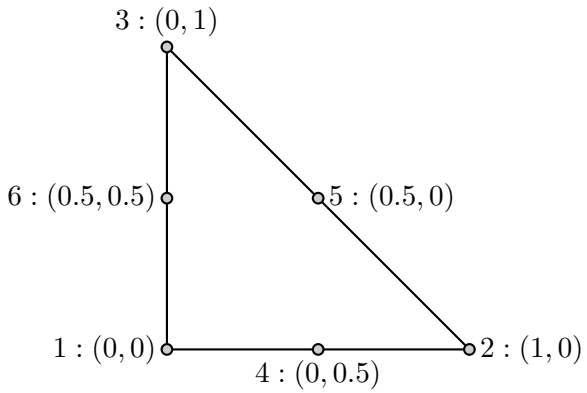
Obrázek 5.3: Závislost velikosti chyby numerického řešení na velikosti kroku sítě.

hustotu $\rho = 1$. Kontrolní matice hmotnosti $\mathbf{M}_{e1}$ byla spočítána v programu Mathematica.

$$\mathbf{M}_{e1} = \frac{1}{360}\begin{pmatrix} 6 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & -4 & 0 & 0 & 0 \\ 0 & 6 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & -4 & 0 & 0 \\ -1 & 0 & 6 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -4 & 0 \\ 0 & -1 & 0 & 6 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -4 \\ -1 & 0 & -1 & 0 & 6 & 0 & -4 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 6 & 0 & -4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -4 & 0 & 32 & 0 & 16 & 0 & 16 & 0 \\ 0 & 0 & 0 & 0 & 0 & -4 & 0 & 32 & 0 & 16 & 0 & 16 \\ -4 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 32 & 0 & 16 & 0 \\ 0 & -4 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 32 & 0 & 16 \\ 0 & 0 & -4 & 0 & 0 & 0 & 16 & 0 & 16 & 0 & 32 & 0 \\ 0 & 0 & 0 & -4 & 0 & 0 & 0 & 16 & 0 & 16 & 0 & 32 \end{pmatrix}$$

Pro výpočet chyby mezi kontrolní maticí $\mathbf{M}_{e1}$ a maticí $\widetilde{\mathbf{M}}_{e1}$ spočítanou programem je zvolena Frobeniova norma z rozdílu obou matic $\|\mathbf{err}\|_F$. Frobeniova norma pro matici je definována vztahem

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}, \tag{5.1}$$



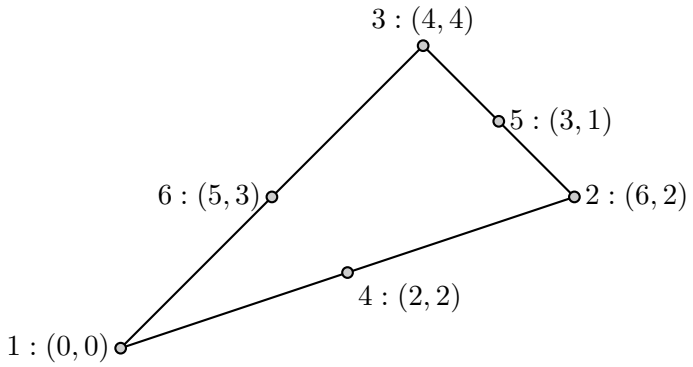
Obrázek 5.4: Poloha uzlů v trojúhelníku $e1$

kde a_{ij} je prvek matice $\mathbf{A}$. Pro tento případ je chyba

$$\|\mathbf{err}\|_F \doteq 5.1089 \cdot 10^{-15}.$$

5.2.2 Matice tuhosti

V sadě jednotkových testů pro třídu `QuadraticIsoparam` je také test výpočtu matice hmotnosti podle vzorce 3.80. Správnost sestavení matice tuhosti jsem testoval porovnáním s dvěma ukázkovým příklady z přednášky [6]. Pokud testovaný element není zdeformovaný, tzn. uzel 4 leží v polovině spojnice uzlu 1 a 2, uzel 5 leží v polovině spojnice uzlu 2 a 3 a uzel 6 leží v polovině spojnice uzlu 3 a 1, je výpočet pomocí 9 Gaussových uzlů přesný, protože integrovaná funkce je polynom menšího stupně než 9.



Obrázek 5.5: Poloha uzlů v trojúhelníku $e2$

První příklad: Pro trojúhelník zobrazený na obrázku 5.5, Youngův modul pružnosti $E =$

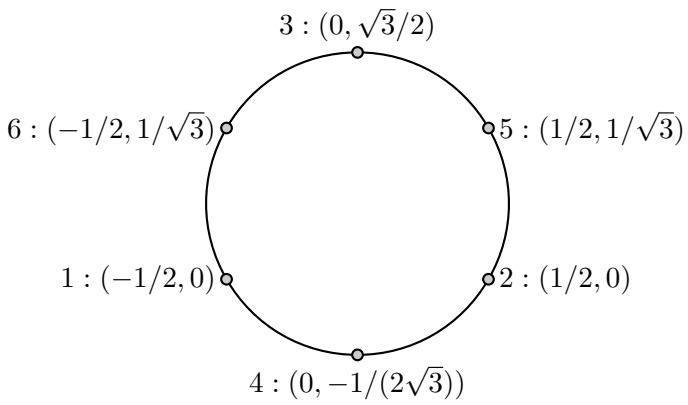
288 a Poissonův koeficient $\nu = \frac{1}{3}$, má podle [6] kontrolní matice tuhosti $\mathbf{K}_{e2}$ tento tvar

$$\mathbf{K}_{e2} = \begin{pmatrix} 54 & 27 & 18 & 0 & 0 & 9 & -72 & 0 & 0 & 0 & 0 & -36 \\ 27 & 54 & 0 & -18 & 9 & 36 & 0 & 72 & 0 & 0 & -36 & -144 \\ 18 & 0 & 216 & -108 & 54 & -36 & -72 & 0 & -216 & 144 & 0 & 0 \\ 0 & -18 & -108 & 216 & -36 & 90 & 0 & 72 & 144 & -360 & 0 & 0 \\ 0 & 9 & 54 & -36 & 162 & -81 & 0 & 0 & -216 & 144 & 0 & -36 \\ 9 & 36 & -36 & 90 & -81 & 378 & 0 & 0 & 144 & -360 & -36 & -144 \\ -72 & 0 & -72 & 0 & 0 & 0 & 576 & -216 & 0 & -72 & -432 & 288 \\ 0 & 72 & 0 & 72 & 0 & 0 & -216 & 864 & -72 & -288 & 288 & -720 \\ 0 & 0 & -216 & 144 & -216 & 144 & 0 & -72 & 576 & -216 & -144 & 0 \\ 0 & 0 & 144 & -360 & 144 & -360 & -72 & -288 & -216 & 864 & 0 & 144 \\ 0 & -36 & 0 & 0 & 0 & -36 & -432 & 288 & -144 & 0 & 576 & -216 \\ -36 & -144 & 0 & 0 & -36 & -144 & 288 & -720 & 0 & 144 & -216 & 864 \end{pmatrix}$$

Pro výpočet chyby mezi kontrolní maticí $\mathbf{K}_{e2}$ a maticí $\widetilde{\mathbf{K}}_{e2}$ spočítanou programem je zvolena Frobeniova norma z rozdílu obou matic $\|\mathbf{err}\|_F$. Pro tento případ je chyba

$$\|\mathbf{err}\|_F \doteq 3.0437 \cdot 10^{-11}.$$

Pokud uzly 4-6 neleží v polovině stran trojúhelníku (123), stane se z polynomiální funkce racionální a ta se dá spočítat pomocí Gaussových uzlů jen přibližně.



Obrázek 5.6: Poloha uzlů v trojúhelníku $e3$

Druhý příklad: Pro trojúhelník zobrazený na obrázku 5.6, Youngův modul pružnosti $E = 504$, Poissonův koeficient $\nu = 0$, má podle [6] kontrolní matice tuhosti $\mathbf{K}_{e2}$ tento tvar[1]

$$\mathbf{K}_{e3} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \end{pmatrix}$$

[1]Protože matice je příliš široká, je její výpis rozdělen na dvě části.

$$\mathbf{A} = \begin{pmatrix} -432.1 & -117.2 & -190.2 & -29.10 & -273.8 & -40.61 \\ 50.79 & -182.7 & -29.10 & -156.6 & -208.6 & -341.0 \\ -432.1 & 117.2 & -273.8 & 40.61 & -190.2 & 29.10 \\ -50.79 & -182.7 & 208.6 & -341.0 & 29.10 & -156.6 \\ -139.8 & 0 & -216.3 & 175.4 & -216.3 & -175.4 \\ 0 & -207.0 & 7.407 & -398.5 & -7.407 & -398.5 \\ 723.6 & 0 & 140.2 & -117.2 & 140.2 & 117.2 \\ 0 & 562.6 & -117.2 & 4.884 & 117.2 & 4.884 \\ 140.2 & -117.2 & 602.8 & -69.71 & -62.78 & 0 \\ -117.2 & 4.884 & -69.71 & 683.3 & 0 & 207.9 \\ 140.2 & 117.2 & -62.78 & 0 & 602.8 & 69.71 \\ 117.2 & 4.884 & 0 & 207.9 & 69.71 & 683.3 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} 661.9 & 158.5 & 141.7 & 21.00 & 92.53 & 7.407 \\ 158.5 & 478.8 & -21.00 & 76.13 & 49.41 & 125.3 \\ 141.7 & -21.00 & 661.9 & -158.5 & 92.53 & -7.407 \\ 21.00 & 76.13 & -158.5 & 478.8 & -49.41 & 125.3 \\ 92.53 & 49.41 & 92.53 & -49.41 & 387.3 & 0 \\ 7.407 & 125.3 & -7.407 & 125.3 & 0 & 753.4 \\ -432.1 & 50.79 & -432.1 & -50.79 & -139.8 & 0 \\ -117.2 & -182.7 & 117.2 & -182.7 & 0 & -207.0 \\ -190.2 & -29.10 & -273.8 & 208.6 & -216.3 & 7.407 \\ -29.10 & -156.6 & 40.61 & -341.0 & 175.4 & -398.5 \\ -273.8 & -208.6 & -190.2 & 29.10 & -216.3 & -7.407 \\ -40.61 & -341.0 & 29.10 & -156.6 & -175.4 & -398.5 \end{pmatrix}$$

Pro výpočet chyby mezi kontrolní maticí $\mathbf{K}_{e3}$ a maticí $\widetilde{\mathbf{K}}_{e3}$ spočítanou programem je zvolena Frobeniova norma z rozdílu obou matic $\|\mathbf{err}\|_F$. Pro tento případ je chyba

$$\|\mathbf{err}\|_F \doteq 18.6461.$$

Matice $\mathbf{K}_{e3}$ byla v [6] spočítána numericky pomocí sedmi Gaussových uzlů. Program jFEM však počítá pomocí devíti uzlů. To je pravděpodobně příčina, proč je v tomto případě chyba mnohem vetší než pro matici $\mathbf{K}_{e2}$.

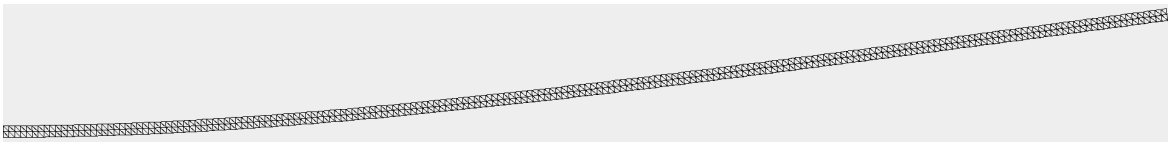
5.2.3 Průhyb nosníku zatíženého spojitým zatížením

Testování bylo provedeno na nosníku s konstantním průřezem obdélníkového tvaru. Parametry nosníku: $l = 10$, $h = 0.1$, $b = 1$, Youngův modul pružnosti $E = 2$, Poissonův koeficient $\nu = 0$. Nosník je na levé straně zafixován v x a levý dolní roh nosníku je zafixován i v ose y. Spojité zatížení působí kolmo na spodní stranu nosníku a bylo spočítáno tak, aby teoretické prohnutí konce nosníku bylo 1 ve směru osy y.

$$q = \frac{8EJ_z}{l^4} = \frac{4}{3 \cdot 10^7}$$

Deformace v obecném místě nosníku je dána vztahem

$$y(x) = \frac{q}{8EJ_z}\left(2l^2x^2 - \frac{4lx^3}{3} + \frac{x^4}{3}\right), \tag{5.2}$$



Obrázek 5.7: Nosník zatížený spojitým zatížením

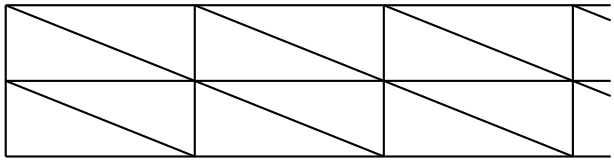
počet dílků	h	$\|\|err\|\|$
10	1	$1,08 \cdot 10^{-2}$
20	0,5	$2,17 \cdot 10^{-3}$
40	0,25	$1,56 \cdot 10^{-4}$
80	0,125	$2,19 \cdot 10^{-4}$

Tabulka 5.2: Závislost velikosti chyby numerického řešení na délce dílku ve směru osy x.

kde x je vzdálenost od pevného konce nosníku, l je délka nosníku, q je spojité zatížení, E je Youngův modul pružnosti v tahu a J_z je kvadratický moment průřezu v ose z, který je pro obdélníkový průřez dán vztahem

$$J_z = \frac{bh^3}{12},$$

kde h je výška nosníku (ve směru osy y) a b je tloušťka nosníku (ve směru osy z).



Obrázek 5.8: Naznačení struktury sítě pro výpočet vetknutého nosníku

Chyba oproti teoretickému posunutí byla zjišťována na neutrální ose nosníku rovnoběžné s osou x. Pro testování konvergence byla vytvořena pravidelná síť tvořená pravoúhlými trojúhelníky podobně jako je znázorněna na obrázku 5.8. Pro první výpočet byl nosník ve směru osy x diskretizován na 10 dílků s ekvidistantním rozestupem a v každém následujícím na dvojnásobný počet, tzn síť byla zjemňována pouze ve směru osy x. L_2 norma chyby je počítána podle vzorce

$$\|err\| = \sqrt{\int_{\Gamma_x} (u - \tilde{u})^2 \, \mathrm{d}\Gamma_x} = \sqrt{\sum_i^{n_e} \int_{\Gamma_{ex}} (u - \tilde{u})^2 \, \mathrm{d}\Gamma_{ex}},$$

kde Γ_x je neutrální osa nosníku, Γ_{ex} je jeden dílek na neutrální osa a n_e je počet dílků na neutrální ose. Integrál v druhé části výrazu je aproximován tak, jako ostatní integrály nad hranou elementu pomocí Gaussových bodů. Výsledky jsou shrnuty do tabulky 5.2. Sloupec h udává délku dílku sítě ve směru osy x.

Průhyb nosníku spočítaný programem *jFEM* konverguje k teoretickému průběhu pouze při zjemňování velmi hrubé sítě. Od jistého okamžiku se chyba přestane zmenšovat a naopak

začne růst. Vypadá to, že posunutí způsobené spojitým zatížením konverguje pro toto zadání k rovnici

$$y(x) = 0.0000333295x^4 - 0.0013332471x^3 + 0.0199985552x^2 + 0.0000176777x$$

i když by měla konvergovat k rovnici 5.2, která po dosazení konkrétních hodnot pro tento příklad má tvar

$$\begin{aligned} y(x) &= \frac{\frac{4}{3\cdot 10^7}}{8\cdot 2\cdot \frac{0.1^3}{12}}\left(2\cdot 10^2x^2 - \frac{4\cdot 10x^3}{3} + \frac{x^4}{3}\right) \\ &= \frac{x^4}{30000} - \frac{x^3}{750} + \frac{x^2}{50} \\ &\doteq 0.0000333333x^4 - 0.00133333x^3 + 0.02x^2, \end{aligned}$$

kde x je vzdálenost od pevného konce nosníku. Nutno však podotknout, že analytické řešení dané vztahem (5.2) vychází z 1D modelu vetknutého nosníku, avšak výše uvedené aproximované řešení je řešením, které vzniklo z 2D modelu přičemž nosník byl ve směru osy y (tedy příčně) diskretizován jak naznačuje obrázek 5.8.

5.3 DynamicFEM

5.3.1 Testování proti analytickému řešení

Pro otestování dynamického mechanického modelu byl zvolen jeden analyticky řešitelný příklady: podélné kmitání tenké tyče připevněné pouze za jeden konec.

5.3.1.1 Podélné kmity tenké tyče

Řešení úlohy je převzato z [3] na straně 39.

Zadání této úlohy včetně okrajových a počátečních podmínek vypadá následovně:

$$\frac{\partial^2 u}{\partial t^2} = v^2\frac{\partial^2 u}{\partial x^2} \qquad x \in (0, L), t \in (0, \infty) \tag{5.3}$$

$$u(x, 0) = \varphi(x), \quad \frac{\partial u}{\partial t}(x, 0) = 0, \qquad x \in (0, L), \tag{5.4}$$

$$u(0, t) = 0, \quad \frac{\partial u}{\partial x}(L, t) = 0, \qquad t \in (0, \infty). \tag{5.5}$$

kde $v = \sqrt{\frac{E}{\rho}}$, E je Youngův modul pružnosti, ρ je hustota materiálu a $\varphi(x)$ udává počáteční polohu tyče. Hledáme funkci podélného posunutí $u(x, t)$, která je funkcí polohy ve směru osy x a času t. Tuto funkci hledáme pro všechny body tyče délky L.

Podle [3] vyhovují zadané úloze funkce ve tvaru

$$u(x, t) = \sum_{n=1}^{\infty} a_n \cos\left(\frac{v(2n-1)\pi}{2L}t\right)\cdot \sin\left(\frac{(2n-1)\pi}{2L}x\right). \tag{5.6}$$

Koeficienty a_n odpovídají koeficientům sinusové řady pro funkci $\varphi(x)$ na intervalu $(0, L)$.

$$a_n = \frac{2}{L} \int_0^L \varphi(x) \sin\left(\frac{(2n-1)\pi}{2L} x\right) \mathrm{d}x \tag{5.7}$$

Pokud si jako funkci $\varphi(x)$, která předepisuje počáteční posunutí, zvolíme funkci

$$\varphi(x) = k_n \sin\left(\frac{(2n-1)\pi}{2L} x\right), \tag{5.8}$$

kde k_n je reálné číslo, bude tyč kmitat pouze v tomto n-tém módu. Rovnice 5.6 se zjednoduší na tvar

$$u(x,t) = k_n \cos\left(\frac{v(2n-1)\pi}{2L} t\right) \cdot \sin\left(\frac{(2n-1)\pi}{2L} x\right). \tag{5.9}$$

Pro porovnání výsledků z programu s teoretickým předpokladem byl porovnán pouze počet kmitů pravého konce nosníku za dobu T_1. Délka periody je

$$T_n = \frac{4L}{v(2n-1)} \tag{5.10}$$

a počet kmitů za dobu T_1 je

$$p_n = \frac{T_1}{T_n} = 2n - 1. \tag{5.11}$$

Testování bylo provedeno na tyči s konstantním průřezem obdélníkového tvaru. Parametry tyče: $l = 10$, $h = 0.1$, $b = 1$, Youngův modul pružnosti $E = 2$, Poissonův koeficient $\nu = 0$. Tyč je zafixována na levé straně v ose x i v ose y. Délka periody pro první mód je $T_1 = 20$.

Na grafech 5.9a až 5.9d je zobrazeno posunutí pravého konce tyče ve směru osy x za dobu T_1 pro módy 1, 2, 3 a 4. Je vidět, že konec tyče kmitá dle teoretického předpokladu.

5.3.2 Porovnání s programem COMSOL

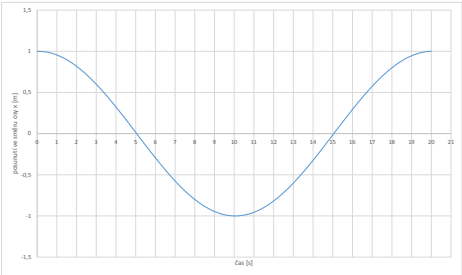
5.3.2.1 Kmitání nosníku namáhaného na tah

Uvažujme následující zadání zobrazené na obrázku 5.10. Testování bylo provedeno na nosníku s konstantním průřezem obdélníkového tvaru. Parametry nosníku: délka $l = 10^{-5}$, výška $h = 5 \cdot 10^{-6}$, šířka $b = 5 \cdot 10^{-6}$, Youngův modul pružnosti $E = 10^5$, Poissonův koeficient $\nu = 0$ a hustota materiálu $\rho = 2330$. Nosník je namáhán vnější sílou $F = 1.2242 \cdot 10^{-7}$ na pravé straně, plošnou sílu získáme tak, že $F_S = \frac{F}{bh}$. Nosník je na levé straně zafixován v ose x i v ose y. Výpočet byl proveden s časovým krokem $\Delta t = 10^{-8}$.

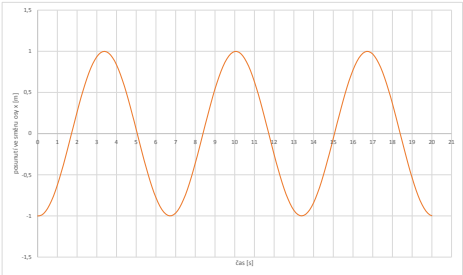
Tato úloha byla porovnána s řešením v programu *COMSOL Multiphysics*. Pro porovnání bylo vyneseno do grafu posunutí ve směru osy x pro uzel, který je umístěn v pravém dolním rohu nosníku. Grafy jsou zobrazeny na obrazcích 5.11a a 5.11b.

Perioda kmitání této soustavy je stejná jako kmitání nezatíženého nosníku v prvním módu tj. podle (5.10)

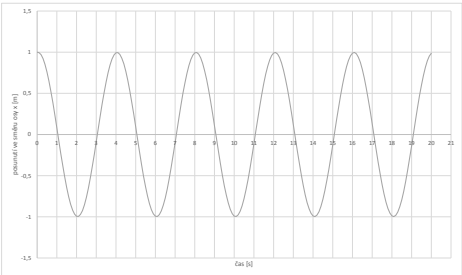
$$T_1 = \frac{4l}{v} \doteq 6.1057 \cdot 10^{-6},$$



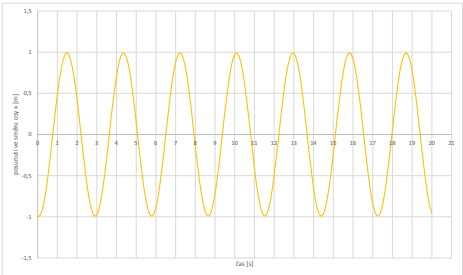
(a) Průběh posunutí konce tyče pro 1. mód



(b) Průběh posunutí konce tyče pro 2. mód

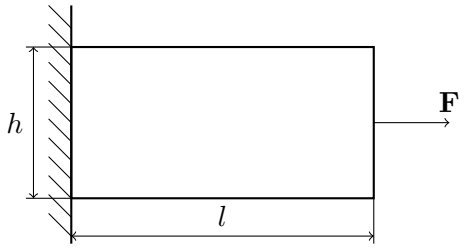


(c) Průběh posunutí konce tyče pro 3. mód



(d) Průběh posunutí konce tyče pro 4. mód

Obrázek 5.9: Kmitání tyče ve zvoleném módu



Obrázek 5.10: Schéma zadání nosníku namáhaného na tah

kde $v = \sqrt{\frac{E}{\rho}}$. Maximální posunutí konce nosníku je

$$u(l) = \frac{2F_S}{E} l \doteq 0.97936 \cdot 10^{-6}$$

Druhý test má stejné zadání jako v předchozím případě až na Poissonův koeficient $\nu = 0.3$. Pro porovnání bylo opět vyneseno do grafu posunutí ve směru osy x pro uzel, který je umístěn v pravém dolním rohu nosníku.

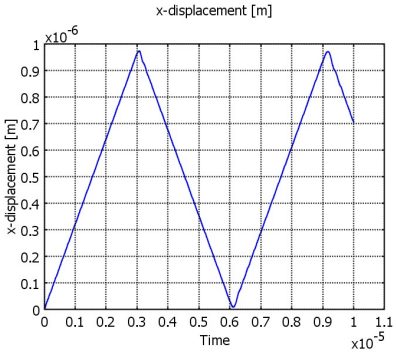
Grafy jsou zobrazeny na obrazcích 5.11c a 5.11d. Z průběhu obou grafů je vidět, že program *jFEM* spočetl téměř shodné hodnoty jako program *COMSOL*.

5.4 ActuatorFEM

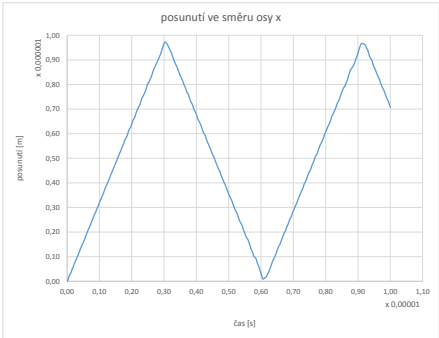
5.4.1 Kmitání nosníku buzeného elektrostatickým polem

Uvažujme následující zadání zobrazené na obrázku 5.12. Testování bylo provedeno na nosníku s konstantním průřezem obdélníkového tvaru. Parametry nosníku: délka $l = 10^{-5}$, výška $h = 50 \cdot 10^{-6}$, šířka $b = 5 \cdot 10^{-6}$, Youngův modul pružnosti $E = 10^5$, Poissonův koeficient $\nu = 0$ a hustota materiálu $\rho = 2330$. Vpravo od nosníku je ve vzdálenosti $d_0 = 5 \cdot 10^{-6}$ umístěna nehybná elektroda. Napětí mezi elektrodou a nosníkem je 150. Nosník je na pravé straně zafixován v ose x i ose y. Výpočet byl proveden s časovým krokem $\Delta t = 10^{-8}$.

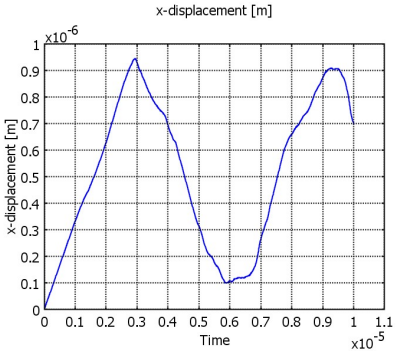
Tato úloha byla porovnána s řešením v programu *COMSOL Multiphysics*. Pro porovnání bylo vyneseno do grafu posunutí ve směru osy x uzlu, který je umístěn uprostřed pravé stěny nosníku. Grafy jsou zobrazeny na obrazcích 5.13a a 5.13b. Z průběhu obou grafů je vidět, že program *jFEM* spočetl velmi podobné hodnoty jako program *COMSOL*.



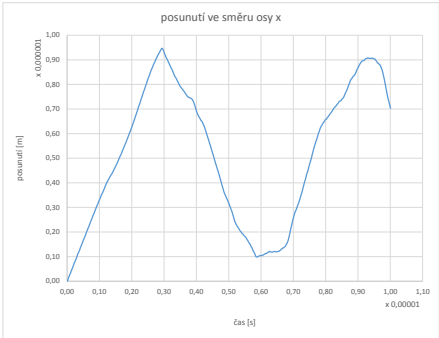
(a) Průběh posunutí koncového budou nosníku v čase spočítaný v programu *COMSOL*, $\nu = 0$



(b) Průběh posunutí koncového budou nosníku v čase spočítaný v programu *jFEM*, $\nu = 0$

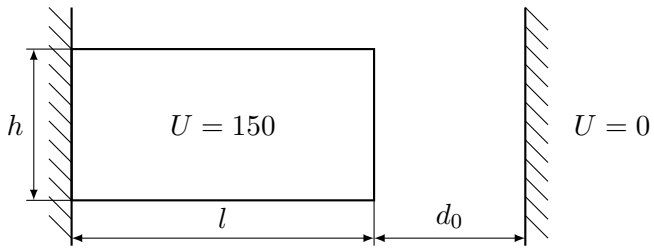


(c) Průběh posunutí koncového budou nosníku v čase spočítaný v programu *COMSOL*, $\nu = 0.3$

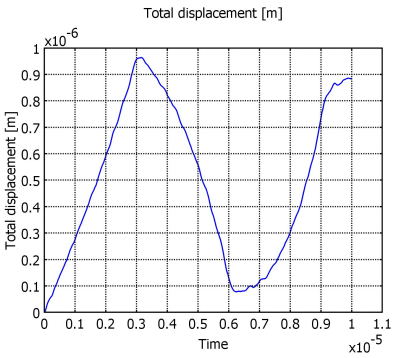


(d) Průběh posunutí koncového budou nosníku v čase spočítaný v programu *jFEM*, $\nu = 0.3$

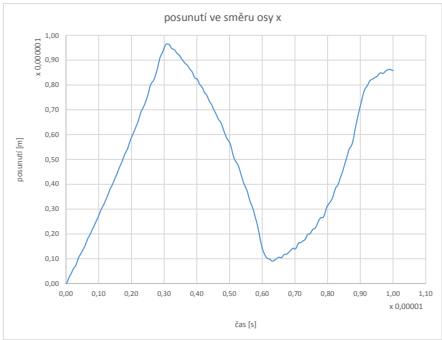
Obrázek 5.11: Kmitání nosníku namáhaného na tah



Obrázek 5.12: Schéma zadání nosníku namáhaného elektrostatickou silou



(a) Průběh posunutí koncového budou nosníku namáhaného elektrostatickou silou v čase spočítaný v programu *COMSOL*, $h = 50 \cdot 10^{-6}$



(b) Průběh posunutí koncového budou nosníku namáhaného elektrostatickou silou v čase spočítaný v programu *jFEM*, $h = 50 \cdot 10^{-6}$

Obrázek 5.13: Kmitání nosníku namáhaného elektrostatickou silou

Kapitola 6

Závěr

Cílem práce bylo navrhnout model elektrostatického aktuátoru s rozloženými parametry, který bude zohledňovat vzájemnou interakci elektrostatického pole a mechanické struktury a vytvořit implementaci v jazyce Java, která bude tento model počítat pomocí metody konečných prvků.

V teoretické části je popsána metoda konečných prvků ve 2D, její aplikace na výpočet rozložení potenciálu a na výpočet statického i dynamického chování pružného lineárního tělesa. Dále je rozebráno vypočtení sil z elektrostatického pole a deformace konečnoprvkové sítě volného prostoru. Je také popsán jeden typ 2D elementu a jeden typ 1D elementu. Jedná se o 2D šestiuzlový trojúhelník a tříuzlovou úsečku. Zmíněna je i metoda numerické integrace pomocí Gaussova kvadraturního pravidla.

V druhé části diplomové práce je popsán program *jFEM* v jazyce Java, který je schopen počítat chování elektrostatického aktuátoru. Detailně je popsána struktura vstupních souborů a pomocná knihovna-program *jDistMesh*, který zjednodušuje přípravu vstupních souborů a umožňuje generování sítě.

Fungování jednotlivých části programu je testováno porovnáním s analytickým řešením či s výstupy z programu *COMSOL Multiphysics*. Při ověřování výpočtu rozložení elektrostatického potenciálu, byla zjištěna konvergence 2. řádu k analytickému řešení. Při porovnání výsledků s výstupy z programu *COMSOL Multiphysics* program *jFEM* vykazuje velmi dobrou shodu.

Domnívám se, že se mi podařilo splnit zadání práce a program je použitelný k zamýšlenému účelu. V budoucnu by bylo možno program dále rozvíjet a to:

- *Přidáním podpory pro více konečnoprvkových elementů*
- *Podporou výpočtu ve 3D*

Literatura

[1] *JUnit 4* [online]. 2014. [cit. 2014-04-06]. Dostupné z: <http://junit.sourceforge.net/javadoc/>.

[2] *What is MEMS Technology?* [online]. [cit. 2014-04-06]. Dostupné z: <https://www.memsnet.org/mems/what_is.html>.

[3] BOŘIL, J. Fourierova metoda řešení parciálních diferenciálních rovnic [online]. Diplomová práce, Masarykova univerzita, Přírodovědecká fakulta, Brno, 2006 [cit. 2014-05-05]. Dostupné z: <http://is.muni.cz/th/3622/prif_m/>.

[4] CHEW, L. P. *Voronoi Diagram / Delaunay Triangulation* [online]. 2007. [cit. 2014-04-06]. Dostupné z: <http://www.cs.cornell.edu/info/people/chew/Delaunay.html>.

[5] DOBEŠ, J. Numerical algorithms for the computation of steady and unsteady compressible flow over moving geometries - application to fluid-structure interaction. Disertační práce, České vysoké učení technické v Praze, Fakulta strojní, Praha, 2007.

[6] FELIPPA, C. A. *Introduction To Finite Element Methods, Chapter 24: Implementation of iso-P Triangular Elements* [online]. 2004. [cit. 2014-04-06]. Dostupné z: <http://www.colorado.edu/engineering/cas/courses.d/IFEM.d/IFEM.Ch24.d/IFEM.Ch24.pdf>.

[7] FELIPPA, C. A. *Introduction To Finite Element Methods, Chapter 14: The Plane Stress Problem* [online]. 2004. [cit. 2014-04-06]. Dostupné z: <http://www.colorado.edu/engineering/cas/courses.d/IFEM.d/IFEM.Ch14.d/IFEM.Ch14.pdf>.

[8] GOOGLE. *Guava: Google Core Libraries for Java 1.6+* [online]. 2014. [cit. 2014-04-06]. Dostupné z: <https://code.google.com/p/guava-libraries/>.

[9] HALLIDAY, S. *Matrix Toolkits Java* [online]. 2014. [cit. 2014-04-06]. Dostupné z: <https://github.com/fommil/matrix-toolkits-java/>.

[10] HALLIDAY, S. *netlib-java* [online]. 2014. [cit. 2014-04-06]. Dostupné z: <https://github.com/fommil/netlib-java>.

[11] HUSÁK, M. *Mikroakturátory – úvod* [online]. 2008. [cit. 2014-04-06]. Dostupné z: <http://micro.feld.cvut.cz/home/X348MS/prednasky/02%20Mikroakturatory_uvod_MIM.pdf>.

[12] JANOCHA, H. *Actuators: Basics and Applications*, s. 343. SPRINGER, Germany, 2004. ISBN 3-540-61564-4.

[13] MÍKA, S. – PŘIKRYL, P. – BRANDNER, M. *Speciální numerické metody : numerické metody řešení okrajových úloh pro diferenciální rovnice*, 1. vyd. Plzeň : Vydavatelský servis, 2006. Dostupné z: <http://home.zcu.cz/~mika/SNM2/SNM2.pdf>. ISBN 80-86843-13-0.

[14] PERSSON, P.-O. – STRANG, G. *DistMesh - A Simple Mesh Generator in MATLAB* [online]. 2012. [cit. 2014-04-06]. Dostupné z: <http://persson.berkeley.edu/distmesh/>.

[15] PLICHTA, J. Výpočet rozložení potenciálu statického elektrického pole. Bakalářská práce, České vysoké učení technické v Praze, Fakulta elektrotechnická, Praha, 2011.

[16] SHEWCHUK, J. R. *Triangle* [online]. 2005. [cit. 2014-04-06]. Dostupné z: <http://www.cs.cmu.edu/~quake/triangle.html>.

[17] THE APACHE SOFTWARE FOUNDATION. *Apache Log4j^{TM}1.2* [online]. 2014. [cit. 2014-04-06]. Dostupné z: <http://logging.apache.org/log4j/1.2/>.

[18] THE APACHE SOFTWARE FOUNDATION. *Apache Maven* [online]. 2014. [cit. 2014-04-06]. Dostupné z: <http://maven.apache.org/>.

[19] THE MATHWORKS, I. *N-D Delaunay triangulation* [online]. 2012. [cit. 2014-04-06]. Dostupné z: <http://www.mathworks.com/help/matlab/ref/delaunayn.html>.

Příloha A

Seznam použitých zkratek

API Application Programming Interface

CG Conjugate gradients

GMRES Generalized minimal residual using restart

GUI Graphical User Interface

JRE Java Runtime Environment

MEMS mikro-elektro-mechanical-systems

MKP Metoda konečných prvků

PDR Parciální diferenciální rovnice

PDF Portable Document Format

Příloha B

Formát vstupních dat

B.1 .node

```
13  2  7  1
1 0.0 0.0 12.0 0.0 0.0 0.0 0.0 0.0 0.0 70
2 0.0 1.0 12.0 NaN 0.0 0.0 0.0 0.0 0.0 70
3 1.0 0.0 12.0 0.0 0.0 250.0 0.0 0.0 0.0 100
4 1.0 1.0 12.0 0.0 0.0 250.0 0.0 0.0 0.0 100
5 0.5 0.5 12.0 0.0 0.0 0.0 0.0 0.0 0.0 68
6 0.5 1.0 12.0 0.0 0.0 0.0 0.0 0.0 0.0 68
7 0.25 0.75 12.0 0.0 0.0 0.0 0.0 0.0 0.0 68
8 0.75 0.75 12.0 0.0 0.0 0.0 0.0 0.0 0.0 68
9 0.5 0.0 12.0 0.0 0.0 0.0 0.0 0.0 0.0 68
10 0.75 0.25 12.0 0.0 0.0 0.0 0.0 0.0 0.0 68
11 0.25 0.25 12.0 0.0 0.0 0.0 0.0 0.0 0.0 68
12 1.0 0.5 12.0 0.0 0.0 250.0 0.0 0.0 0.0 116
13 0.0 0.5 12.0 NaN 0.0 0.0 0.0 0.0 0.0 70
```

B.2 .ele

```
4  6  1
1 4 2 5 7 8 6 19
2 1 3 5 10 11 9 19
3 3 4 5 8 10 12 19
4 2 1 5 11 7 13 19
```

B.3 .mat

```
1 6
#atributy: ro E nu permitivita thickness isMechanicArea
19 260 57e9 0.19 2.095e-15 50e4 1
```

Příloha C

jDistMesh.js

```
importPackage(Packages.cz.cvut.fel.plichjan.distmesh.inputs);
importPackage(Packages.cz.cvut.fel.plichjan.distmesh.matlab);
importPackage(Packages.cz.cvut.fel.plichjan.distmesh);
importClass(Packages.delaunay.Pnt);
importClass(java.util.ArrayList);

var zero = [0., 0.];
var Constrain = {
    df: null, flags: null, potential: null, displacement: null, bForce:
        null, vForce: null
};

var IS_SET_DISPLACEMENT = 2;
var IS_SET_POTENTIAL = 4;
var IS_SET_FORCE = 8;
var IS_PRINTABLE = 16;
var IS_SET_BOUNDARY_FORCE = 32;
var WALL_NODE = 64;

function callDf(df, node) {
    return df.call(node[0], node[1]);
}

function Inside(df) {
    return {
        call: function (x, y) {
                return Math.max(0., df.call(x,y));
            }

    };
}

function forEvery () { return -1; }

function storeMesh(file, mesh2, h0, list, trList) {
    var nodePrinter = new NodePrinter(file + ".node");
    nodePrinter.printHeader(mesh2.p.length, 2, 7);
    for (var i = 0; i < mesh2.p.length; i++) {
        var node = mesh2.p[i];
```

```
        var flags = 0;
        var potential = 0., displacement = zero, bForce = zero, vForce =
            zero;

        for (var j = 0; j < list.length; j++) {
            var item = list[j];
            if (Math.abs(callDf(item.df, node)) < h0) {
                flags |= item.flags;
                potential = (item.potential || potential);
                displacement = (item.displacement || displacement);
                bForce = (item.bForce || bForce);
                vForce = (item.vForce || vForce);
            }
        }

        nodePrinter.printNode(i+1, node, [
            potential,
            displacement[0], displacement[1],
            bForce[0], bForce[1],
            vForce[0], vForce[1]
        ], flags);
    }
    nodePrinter.close();

    var centroid = Matlab.computeCentroids(mesh2.p, mesh2.t);
    var elePrinter = new ElePrinter(file + ".ele");
    elePrinter.printHeader(mesh2.t.length, 6);
    for (i = 0; i < mesh2.t.length; i++) {
        var tr = mesh2.t[i];
        var matId = 0;

        for (j = 0; j < trList.length; j++) {
            item = trList[j];
            if (callDf(item.df, centroid[i]) < 0) {
                matId = item.matId;
            }
        }
        elePrinter.printEle(i+1, [tr[0]+1, tr[1]+1, tr[2]+1, tr[4]+1, tr
            [5]+1, tr[3]+1], matId);
    }
    elePrinter.close();
}
```

Ukázka kódu C.1: Pomocné funkce, konstanty a struktury v JavaScriptu

Příloha D

Instalační a uživatelská příručka

D.1 Minimální požadavky

Před instalací aplikace *jFEM* a aplikace *jDistMesh* na cílový počítač je třeba se ujistit, že jsou splněny následující požadavky:

- operační systém s grafickým uživatelským rozhraním umožňující spuštění Java aplikací.
- nainstalované JRE 6 Update 45

D.2 Instalace obou aplikací

Aplikaci *jFEM* a zároveň aplikaci *jDistMesh* nainstalujeme tak, že nakopírujeme obsah adresáře `bin` na požadované místo v cílovém počítači. Tím je celý instalační proces ukončen.

D.3 Spuštění a ovládání aplikace jDistMesh

Aplikaci *jDistMesh* spustíme příkazem `jDistMesh.cmd` z konzole libovolného operačního systému. Poté vybereme v menu `File` položku `Open`. Tím se nám otevře dialog na výběr soubor vstupního skriptu. Vybereme si požadovaný soubor a volbu potvrdíme stiskem tlačítka `Open`.

Po této operaci program provede zvolený skript. Průběh generování sítě je vidět v okně aplikace. Pokud síť odpovídá naší představě, program zavřeme vybráním položky `Close` v menu `File` nebo kliknutím na systémové zavírací tlačítko (v MS Windows je to křížek v pravém horním rohu).

D.4 Spuštění a ovládání aplikace jFEM

Soubor `jFEM.jar` obsahuje čtyři různé aplikace popsané v části 4.3. Jejich spuštění je popsáno v následujících částech.

D.4.1 ElectrostaticFEM

```
 java -cp jFEM.jar -DonlyPrintable cz.cvut.fel.plichjan.fem.mtj.ElectrostaticFEM
<filename>
```

`<filename>` Cesta ke vstupnímu souboru bez přípony.

D.4.2 MechanicsFEM

```
java -cp jFEM.jar -DonlyPrintable cz.cvut.fel.plichjan.fem.mtj.MechanicsFEM <filename>
```

`<filename>` Cesta ke vstupnímu souboru bez přípony.

D.4.3 DynamicFEM

```
java -cp jFEM.jar -DonlyPrintable cz.cvut.fel.plichjan.fem.mtj.DynamicFEM <filename>
<deltaT> <stepCount> <printStep>
```

`<filename>` Cesta ke vstupnímu souboru bez přípony.

`<deltaT>` Velikost časového kroku v sekundách.

`<stepCount>` Celkový počet kroků simulace.

`<printStep>` Po kolika krocích se bude tisknout stav simulace na standardní výstup.

D.4.4 ActuatorFEM

```
java -cp jFEM.jar -DonlyPrintable cz.cvut.fel.plichjan.fem.mtj.ActuatorFEM <filename>
<deltaT> <stepCount> <printStep>
```

`<filename>` Cesta ke vstupnímu souboru bez přípony.

`<deltaT>` Velikost časového kroku v sekundách.

`<stepCount>` Celkový počet kroků simulace.

`<printStep>` Po kolika krocích se bude tisknout stav simulace na standardní výstup.

Struktura trojice vstupních souboru je popsána v části 4.1.

Příloha E

Obsah přiloženého CD

Na přiloženém CD je struktura adresářů, kterou ukazuje tabulka E.1.

Adresář	**Popis**
`/bin`	adresář s přeloženým programem
`/data`	data získaná v průběhu testování programu
`/install`	instalační soubory Java JRE
`/javadoc`	vygenerovaná dokumentace JavaDoc
`/src`	zdrojové texty programu
`/text`	text bakalářské práce ve formátu PDF
`/text_src`	text práce ve zdrojovém formátu

Tabulka E.1: Obsah přiloženého CD