TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B 2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Generování lokálně zjemněných nestrukturovaných sítí pro výpočet proudění metodou konečných prvků nebo konečných objemů

Generation of locally refined unstructured meshes for finite element or finite volume computations of flow

Bakalářská práce

Autor: **Václav Řidký**
Vedoucí práce: Ing. Petr Šidlof, Ph.D

V Liberci 20. 5. 2009

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé BP a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce.

Datum: 20.5.2009

Podpis:

Poděkování

Chtěl bych poděkovat hlavně vedoucímu této bakalářské práce Ing. Petru Šidlofovi , Ph.D a také i dalším lidem který mi s touto prací pomohli. Jejich rady a zkušenosti mi pomohly ke vzniku této práce. Zároveň také děkuji své rodině za poskytnutou trpělivost a zázemí.

Abstrakt

Cílem této bakalářské práce bylo seznámení se s problematikou generování sítí pro výpočty metodou konečných prvků a objemů. Poté bylo nutné vybrat z velkého množství nabízených generátorů sítí program a využít ho ke generování sítě pro úlohu proudění vzduchu v hlasivkách. Hlavní parametry podle kterých byl vybrán software je lokální zjemňování, možnost výběru typu sítě (trojúhelníková, čtyřúhelníková, atd.) a dále pak záruku rozvoje toho generátoru. Proto byl zvolen generátor GMSH, jedná se o open-□source software. Tento software splňuje veškeré požadavky pro jednoduché a rychlé vytvoření sítě s lokálním zjemněním. Síť vytvořena tímto software bude použita pro vypočet proudění metodu konečných prvku v projektu, který se zabývá problematikou proudění vzduch v hlasivkách. Součástí práce bylo také vytvoření program pro konverzi dat ze souboru *.msh na soubor *.maill.

Klíčová slova: sítě, generátor GMSH, metoda konečných prvků a objemů

The aim of this work was to get familiar with the problem of generating computational meshes for the calculations using the finite element and finite volume methods. It was then necessary to select from a large number of mesh generators available on the internet and use it to generate a mesh for numeric simulation of airflow in human vocal folds. The main requirements on the software was the ability to create locally refined meshes, choose between various element types (triangular, quadrangular, etc.) and stable development community. Finally, an open-source generator GMSH was chosen. This software fulfils all the requirements for easy and quick creation of a locally refined mesh. The mesh will be used for numerical computations of airflow in human vocal folds. Part of this work was also creating a program to convert data from GMSH native format *. msh into format *. maill.

Keywords: mesh, generator GMSH, finite element method and finite volume methods

Obsah

Úvod

Důvod proč jsem si vybral toto zadání práce je ten, že mě zajímá modelování jako celek. Jelikož mé vzdělání v oblasti matematiky nedosahuje ještě tak vysoké úrovně, abych se pokoušel řešit samotnou úlohu, tedy sestavení diferenciálních rovnic a jejich řešení pomocí metody konečných prvků. Zaměřím se tedy na část, která tomuto předchází a jaká úskalí sebou přináší. Své poznatky, které jsem o tomto tématu nastudoval se vám tedy pokusím shrnout ve své práci.

Abych své poznatky a získané informace mohl využít, budu provádět přípravnou fázi pro modelování proudění vzduchu v hlasivkách. Na tomto projektu se již několik let podílí i můj vedoucí Petr Šidlof. Proto bude součástí práce vytvoření několika diskretizačních sítí modelu hlasivky a jejich následné využití pro výpočet.

Modelování proudění vzduchu v hlasivkách má význam hlavně pro oblast medicíny. Pokud bude pomocí různých simulací odhaleno chování hlasivek ve vzduchovém proudu, bude možné vytvořit mechanickou náhradu hlasivky. Tato náhrada pak bude sloužit lidem, kteří mají své vlastní hlasivky poškozeny, což jim umožní významné zkvalitnění života.

Proudění vzduch je z technického hlediska velice komplikované. Nyní se řeší problém chování proudění vzduchu v místě zužovaní hlasivky a v prostoru za tímto zúžením. Na stěnách dochází ke tření a rychlost se v těchto místech snižuje a vzduch tak neproudí v celém prostoru stejnou rychlostí. Změna velikosti kanálu hlasivky vlivem rozevíraní a stahování způsobuje také změnu rychlosti. Čím je kanál užší, tím větší rychlosti vzduch v daném místě dosahuje. To způsobuje, že v oblastech za tímto zúžením vznikají víry a turbulence .

Pro zkoumání těchto dějů právě slouží numerické modelovaní založené na metodě konečných prvků nebo metodě konečných objemů. Numerické modely umožňují pozorovat vývoj rychlosti proudění v celé hlasivce. Kvalita výsledného řešení je pak ovlivňovaná zvolenou diskretizační sítí, hlavně pak velikostí elementů v dané síti. V oblastech velkých změň je třeba, aby velikost elementu byla co nejmenší, aby se dosáhlo co nejpřesnějšího řešení, ale zároveň počet elementu nesmí být příliš vysoký z důvodu výpočetní náročnosti. Proto se využívá lokálního zjemnění, které zaručí, že výsledek bude dostatečně přesný a počet elementů bude nižší, než kdyby se zjemnila celá síť.

2 Metoda konečných prvků

2.1 Vlastnosti

Metoda Konečných prvků se dnes využívá pro řešení mnoha fyzikálních problémů. Jedná se o numerickou metodu, která hledá přibližné řešení. Hlavním principem je rozdělení sledované oblasti na konečněprvkovou síť, kde se prvek této sítě nazývá element. Elementy mají jednoduchý tvar, pro 2D oblast je to například trojúhelník a čtyřúhelník, pro 3D oblast je to například čtyřstěn, šestistěn a hranol. V dalším kroku je nutné převedení řešeného problému na sledované oblasti z parciálních diferenciálních rovnic na soustavu lineárních rovnic (diskrétní problém) vznikne matice *A*. Z konstrukce bázových funkcí (konečných prvků) plyne, že výsledná matice problému je řídká, tj. že většina jejích prvků je nulových. Výsledné řešení takto sestaveného matematického modelu nikdy neodpovídá zcela přesně realitě, jelikož při samotném výpočtu dochází k zaokrouhlení čísel. Zvýšení přesnosti modelu je možné dosáhnout zvýšením počtu elementu nebo zvýšením řádu bázových funkcí.

Hlavním důvodem, proč je metoda konečných prvků v dnešní době tak rozšířená, je její robustnost, přesnost a možnost jí použít na velké množství problémů. Přesnost výpočtu pomocí této metody není ovlivněna vlastnostmi materiálu a problémem, který je řešen. To je jeden z významných parametrů jelikož je v dnešní době požíváno mnoho materiálů s různými vlastnostmi a parametry, a také spektrum řešených problémů je rozsáhlé.

Proto velké množství výpočetních softwarů nabízených na trhu využívá metodu konečných prvků, jako nástroj pro vyřešení problému, který mu uživatel zadá. Výhoda těchto softwaru je, že projektant, který neovládá samotnou metodu konečných prvků , ale rozumí problému, který chce řešit, může problém vyřešit. To vede k zefektivnění práce a snížení nákladu při tvorbě nových věcí. [1]

2.2 Využití

Metoda konečných prvků se využívá při modelování. Modelování je odvětví, které se zabývá tvorbou virtuálních modelů skutečných problémů. Modelování proto nachází uplatnění v různých oborech průmyslu a také ve vědě. Modelování umožňuje daný problém prozkoumat a ověřit jeho chování v závislosti na vnějších podmínkách, času, atd.. Modelování je dnes velice často využíváno jelikož tvorba virtuálního modelu je levnější a rychlejší než tvorby modelu reálného. Virtuální model lze opakovaně testovat

v různých podmínkách. Při nastavení nových podmínek stačí pouze opět spustit výpočet. Modelování také složí k předvídání chování jevů, které trvají velice dlouhou dobu a není je možné tedy testovat v reálném čase.

2.3 Zpracování dat metody konečných prvků

Metoda konečných prvků pro své spuštění potřebuje na vstupu data. Tyto vstupní data obsahují informace o řešené oblasti. Vstupní data pro metodu konečných pak jsou vytvořena v tzv. preprocessingu. Po vyřešení problému je řešení opět zapsáno do souboru, jedná se převážně o číselné hodnoty. Zpracování výstupních dat z metody konečných prvků se pak provádí v tzv. postprocessingu

Na každou tuto část lze požít některý ze široké škály nabízených softwarů nebo využít kompletní balík programů jako je například ANSYS, který obsahuje všechny části včetně výpočtové. Tyto softwary jsou, ale mnohdy drahé, na druhou stranu samotná realizace kompletního modelování včetně vizualizace výsledků, se provádí velice snadno jen v několika krocích. Požitím jednotlivých softwarů vzniká problém s kompatibilitou dat a nutnost je pomocí dodatečně vytvořených konvertorů převádět na soubory potřebné pro další část nebo je nutné vybrat takové softwary, které kompatibilitu jednotlivých dat zajistí. To přináší další úskalí a to nutnost danou problematiku dobře chápat.

Preprocessing se skládá ze tří operací a to vytvoření geometrie, diskretizační sítě a zadání okrajových podmínek. Preprocessing bude vysvětlen a důkladně rozpracován v následující kapitole, jelikož se jedná o hlavní část bakalářské práce.

Postprocessing převede vypočtené hodnoty na grafickou formu. Ve které je velikost zobracené veličina rozlišena barevným spektrem, doplněná o měřítko, ze kterého lze odečíst jaká hodnota odpovídá dané barvě. Výrazné barvy se nejčastěji používají pro nejvyšší hodnoty. Grafické zobrazení je pak mnohem lépe názorné něž jen ohromné množství čísel.

3 Preprocessing

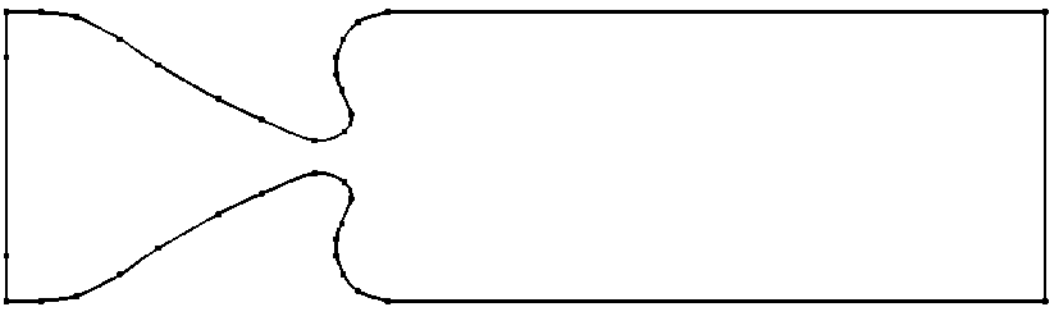
3.1 Úvod

Na tuto část se zaměřím více dopodrobna, jelikož se jedná o jeden ze stěžejních bodů mé bakalářské práce. Proto se tématu Preprocesingu budu věnovat ve dvou kapitolách, tyto kapitoly nesou název Preprocessing a Diskretizační síť.

Jak již bylo zmíněno v předchozí kapitole preprocessing je rozdělena do tří částí. První část je tvorba geometrie řešeného problému, nejčastěji se požívá geometrie 2D a nebo 3D. Druhá část je generování sítě, to v praxi znamená, že geometrie, která reprezentuje řešenou oblast, je rozdělena na konečný počet malých elementů. Třetí část je zadávání okrajových podmínek, tuto část jsme ve své práci neprováděl.

3.2 Geometrie modelu

Základní elementy používané v geometrii modelu jsou bod, úsečka, kružnice a obecná křivka proložená body (spline). Některé generátory umožňuji vložit geometrii vytvořenou pomocí rýsovacích programů CAD. To se především uplatňuje u složitých tvarů, jako je například blok motoru nebo prototyp nových zařízení (automobil, letadlo, atd.) . Vytvořená geometrie může být plně 3D nebo jako řez (2D). V generátoru sítí GMSH je tvorba geometrie matematického modelu jako jeden ze základních modulů (bloků). Výsledná geometrie je uložena do souboru s příponou *.geo. Tvar hlasivky vytvořený v tomto generátoru (viz obr. 3.1). Celá geometrie je vytvořená za pomoci tří prvků, kterými jsou bod, úsečka, křivka proložená body (spline). Rozměry pro daný model jsou získány z měření, které bylo součástí celého projektu modelování proudění vzduchu v lidských hlasivkách a byly dodány společně se zadáním. Výsledný tvar zakřivení samotné hlasivky je tvořen křivkou proloženou mezi body. Z měření bylo získáno několik tisíc bodů, ale pro přehlednost jich bylo vybráno 13, to stačilo pro získání požadovaného tvaru. [2]

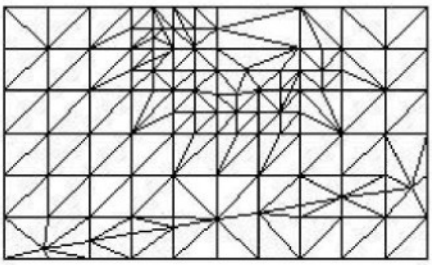


Obr 3.1: Základní tvar matematického modelu hlasivky

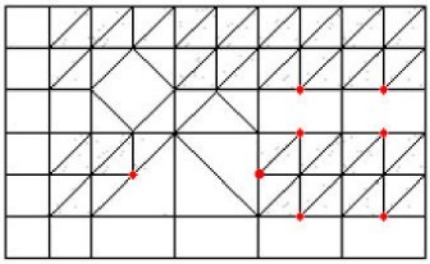
4 Diskretizační síť

4.1 Základní vlastnosti

Diskretizační síť je základem metody konečných prvků. Rozděluje geometrii řešeného problému na elementy. Základní rozdělení sítí je na síť izotropní a lokálně zjemněnou. Izotropní síť lze chápat jako síť, ve které mají všechny elementy stejnou velikost. Lokálně zjemněná síť je pak síť obsahující elementy o různých velikost, je však nutné dodržet poměr mezi velikostí největšího a nejmenšího elementu. Pokud na toto zapomeneme můžeme tím znehodnotit výsledek výpočtu. Důležité je také dodržet, aby výsledná síť byla konformní, nesmí obsahovala volné uzly. Nekonformní sítě lze využít jen pro speciálně upravené výpočetní algoritmy. Ukázky konformní (viz obr. 4.1) a nekonformní sítě (viz obr. 4.2).[3]

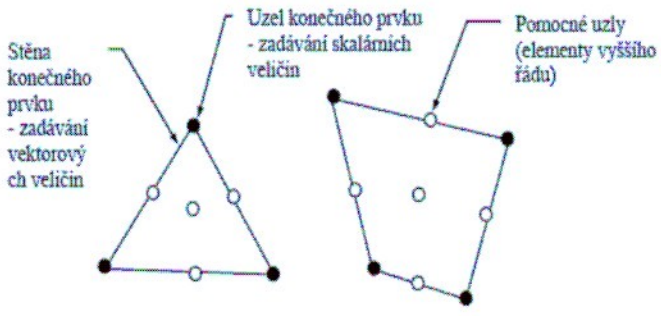


Obr. 4.1: Konformní síť (převzato z [3])



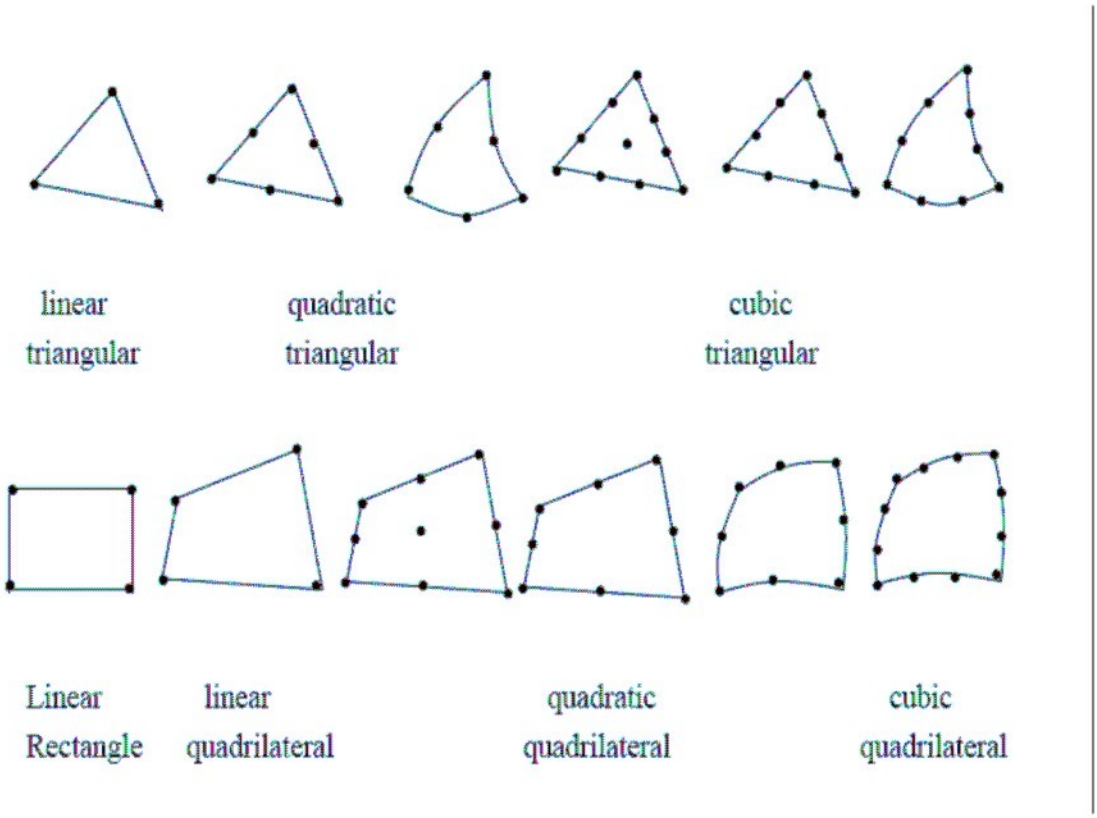
Obr. 4.2: Nekonformní síť s volnými (visícími uzly) (převzato z [3])

S rostoucím počtem elementů se zvyšuje výpočetní náročnost, protože roste počet neznámých, které se řeší. Uzly jsou definovány pomocí souřadnic v dané geometrii. Jeden uzel může být společný více elementům. Uzly rozdělujeme do skupin. První skupinu tvoří uzly hlavní nazývané uzly konečného prvku, jedná se o vrcholové uzly na elementu. Druhou skupinu tvoří uzly pomocné (viz obr. 4.3). Tyto jsou umístěny na hranách, stěnách a uvnitř elementu.



Obr. 4.3: Elementy a uzly [4]

Elementy, které jsou popsány i pomocnými uzly, se nazývají elementy vyššího řádu. Elementy vyššího řádu jsou kvadratické, kubické a atd. , s rostoucím počtem uzlů se zvyšuje řád. Příklad trojúhelníkový element, který je definován pomocí devíti uzlů je element třetího řádu, ale mnohem častěji se využívá označení kubický. Elementy vyššího řádu umožňují získat přesnější řešení s menším počtem elementů. Možnosti elementů prvního řádu (lineární) až element třetího řadu (viz. obr. 4.4).

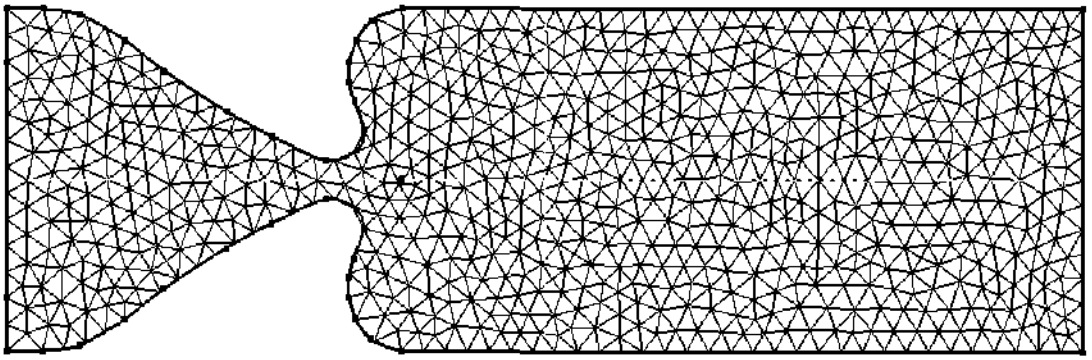


Obr. 4.4: Typy lineárních elementů a elementů vyššího řádu [5]

Jednotlivé elementy k sobě musí přiléhat vždy celými stěnami (3D) nebo celými hranami (2D). Hrany a plochy jednotlivých elementů mohou být rovné nebo zakřivené. Zakřivené hrany a plochy se však využívají ve zcela výjimečných případech. Důležitým aspektem pro přesnost řešení je, aby ve vygenerované síti nevznikaly elementy nevhodného tvaru. Nevhodné tvary budou zmíněny v kapitolách, ve kterých budou popsány jednotlivé typy elementů. Nevhodný tvar vzniká nejčastěji při hierarchickém zjemňování sítě, ve složitých modelech nebo v oblasti **singularit**. Singularita je s geometrického hlediska bod, kterému lze přiřadit více různých souřadnic. Jedná se o hranu nebo roh uvnitř tělesa.

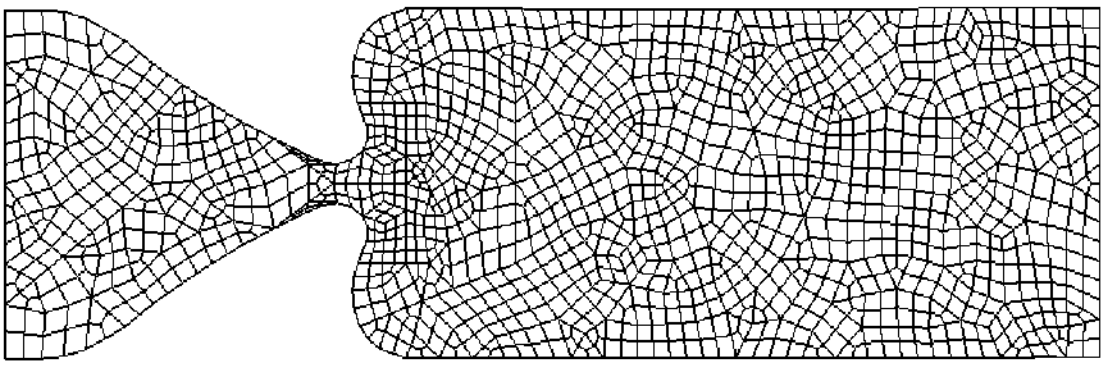
4.2 Elementy diskretizační sítě

Elementy pro 2D geometrii mají tvar trojúhelníku nebo čtyřúhelníku. Trojúhelníková síť se používá nejčastěji příklad trojúhelníkové (triangulační) sítě (viz obr. 4.5). Nevhodné tvary elementů pro trojúhelníkový tvar jsou trojúhelníky ve kterých jsou vnitřní úhly blízké nule. Naopak nejvýhodnější tvar trojúhelníkového elementu je, když se element svým tvarem blíži rovnostrannému trojúhelníku neboli rozdíl velikostí jeho vnitřních úhlů je co nejmenší.



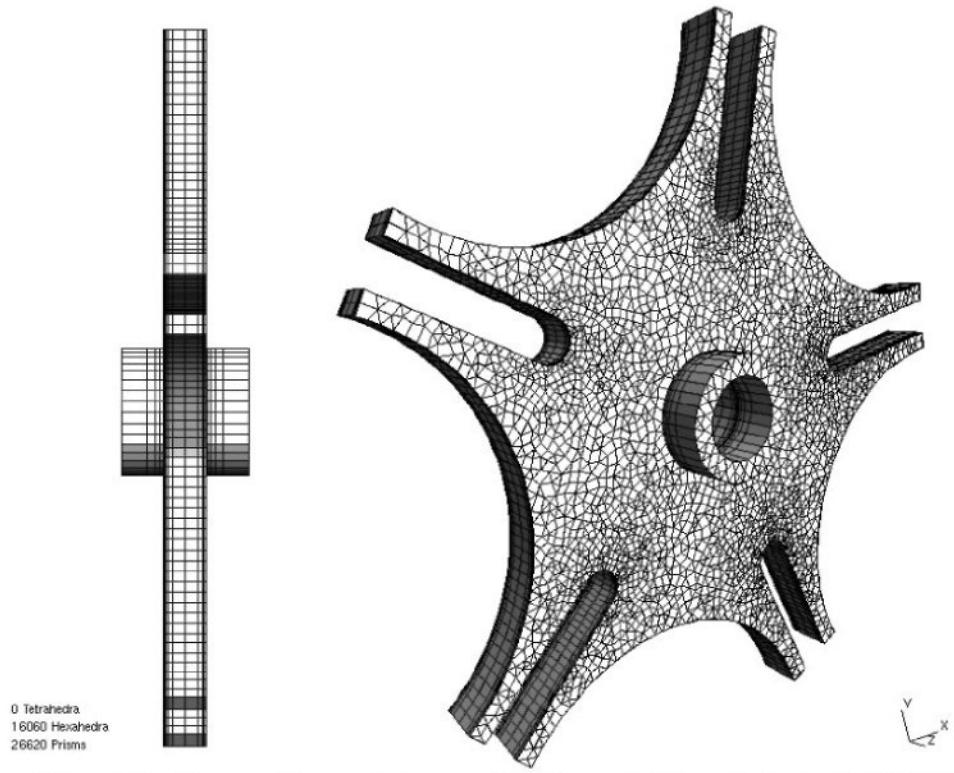
Obr. 4.5: Trojúhelníková sít pro matematický model hlasivek hrubá sít

Čtyřúhelníková síť je také velice často používaná hlavně u strukturovaných sítí, ukázka čtyřúhelníkové sítě (viz.obr. 4.6). U čtyřúhelníků jsou nevhodné tvary, kdy poměr stran je veliké číslo, protáhlý obdélník. Další nevhodný tvar je čtyřúhelník ve tvaru lichoběžníků a to tehdy, když se velikost jeho dvou vnitřních úhlu blíží k nule, protáhlý lichoběžník. [8]



Obr. 4.6: Hlasivka se sítí typu čtyřúhelník

Pro trojrozměrnou geometrii se požívají elementy typu čtyřstěn, osmistěn, pyramida, kvádr. Čtyřstěn lze použít i u velice složitých geometrických tvarů, které obsahují různé výřezy. Osmistěny a kvádry jsou používány pří diskretizaci pravidelných útvarů. Ukázka 3D modelu diskretizovaná pomocí hranolů a osmistěnu (viz. obr. 4.7) . Na obrázku je v levém dolním okraji vidět počet jednotlivých typů elementů. Jak si můžeme všimnout počet elementů není zanedbatelný, ale přitom je síť hrubá. Proto bude následná kapitola věnování problematice zjemňování sítí.



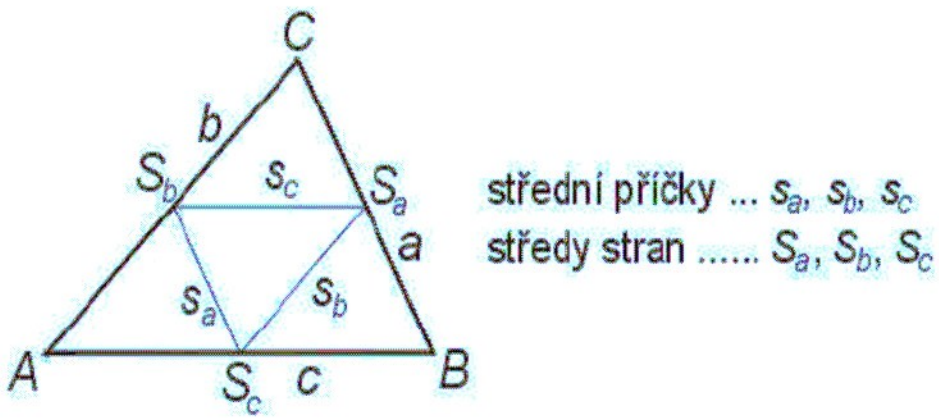
Obr. 4.7: Síť na 3D modelu součástky-vnější maltézský kříž [6]

5 Zjemňování sítě

5.1 Úvod

Pro numerické zpřesnění výsledku lze použít dvě strategie: zjemnění sítě nebo zvýšení řádu elementu. Zjemňování sítě je nutné pro přesnější řešení úlohy. Pokud jsou patrné chyby v řešení už na první pohled, například když se předpokládá hladký průběh funkce dané proměnné a nám simulací vyjde průběh se strmými přechody. Možnost zjemnění jsou dvě základní, **izotropní** (zjemnění celé sítě) a **lokálně zjemněné** (zjemňujeme více v oblastech, kde předpokládáme například velké gradienty řešení)**.**

Příklad správně provedeného zjemnění (viz. obr. 5.1). Zde byl původní trojúhelníkový element rozdělen na čtyři nové trojúhelníkové elementy. Správné rozdělení elementu při zjemňování je důležitá věc, aby nevznikaly elementy nevhodných tvarů. To bylo již zmíněno dříve v kapitole 4.2 .



Obr. 5.1: Trojúhelník ABC rozdělený středními příčkami na trojúhelníky AS_cS_b, S_cS_aB, $S_cS_aS_b$ a S_bS_aC [8]

Na zjemňování sítě se musí pohlížet ze dvou stran. A to rozdělit oblast na co největší počet elementů, ale na druhou stranu je nutné si uvědomit, že řešení úlohy s velkým počtem elementů může trvat neúměrně dlouho. Dalším parametrem kterým je nutné zohlednit je hardwarové vybavení, na kterém bude samotný výpočet probíhat, především z pohledu procesoru a operačních pamětí.

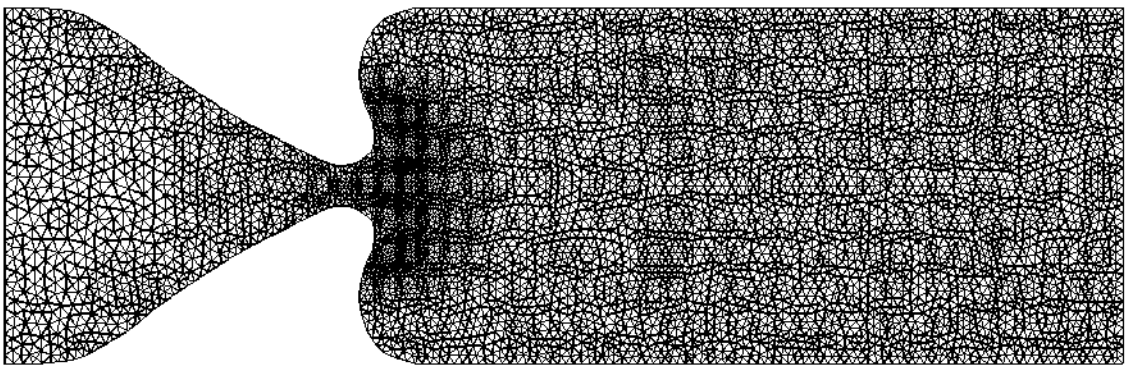
5.2 Zjemňování izotropní sítě

Zjemňování izotropní sítě se provádí velice jednoduše tím, že se zmenší velikost elementu v celém modelu. Počet elementů se při tomto zvyšuje hlavně u 2D modelu kvadraticky a u 3D modelu je to dokonce kubicky. Což znamená pokud dáme velikost elementů poloviční, počet elementů se u 2D modelu zvýší 4-krát a u 3D modelu dokonce 8-krát. A to i v částech úlohy, kde taková hustota sítě není nutná. Počet uzlů se tak velmi rychle dostane do vysokých hodnot, že řešení takovéto úlohy je z časově hlediska nepřijatelné. Z toho důvodů je patrné, že hlavně u 3D modelů je uplatnění této metody nevhodné a využívá se lokálně zjemněné sítě.

5.3 Lokální zjemňováni sítě

Jedná se o síť, která nemá konstantní hustotu elementů. Lokálně zjemněná síť nám umožňuje v určitých úlohách výrazné snížení počtu elementů, bez toho aniž by byla ovlivněna přesnost řešení. Jedná se o úlohy, kde se na řešené oblasti vyskytují místa s velkým gradientem řešené proměně. Jako příklad může vzít proudění vzduch v kanálů, kde k nárůstu gradientu rychlosti dochází v místech zaškrcení kanálu. Pro lepší představu (viz. obr.5.2).

Lokálně zjemněné sítě v sobě skrývají problém v návaznosti elementů s rozlišnou velikostí. To vyžaduje speciální algoritmy, které jsou složitější než algoritmy pro tvorbu izotropních sítí. Je také nutné dodržet, aby výsledná síť byla i po lokálním zjemnění stále konformní (neobsahovala tedy volné uzly). Problematika konformity sítě je již řešena v kapitole 4.1 .



Obr. 5.2: Lokálně zjemněná síť modelu hlasivky

5.4 Vliv zjemnění na výpočetní náročnost a přesnost výsledku

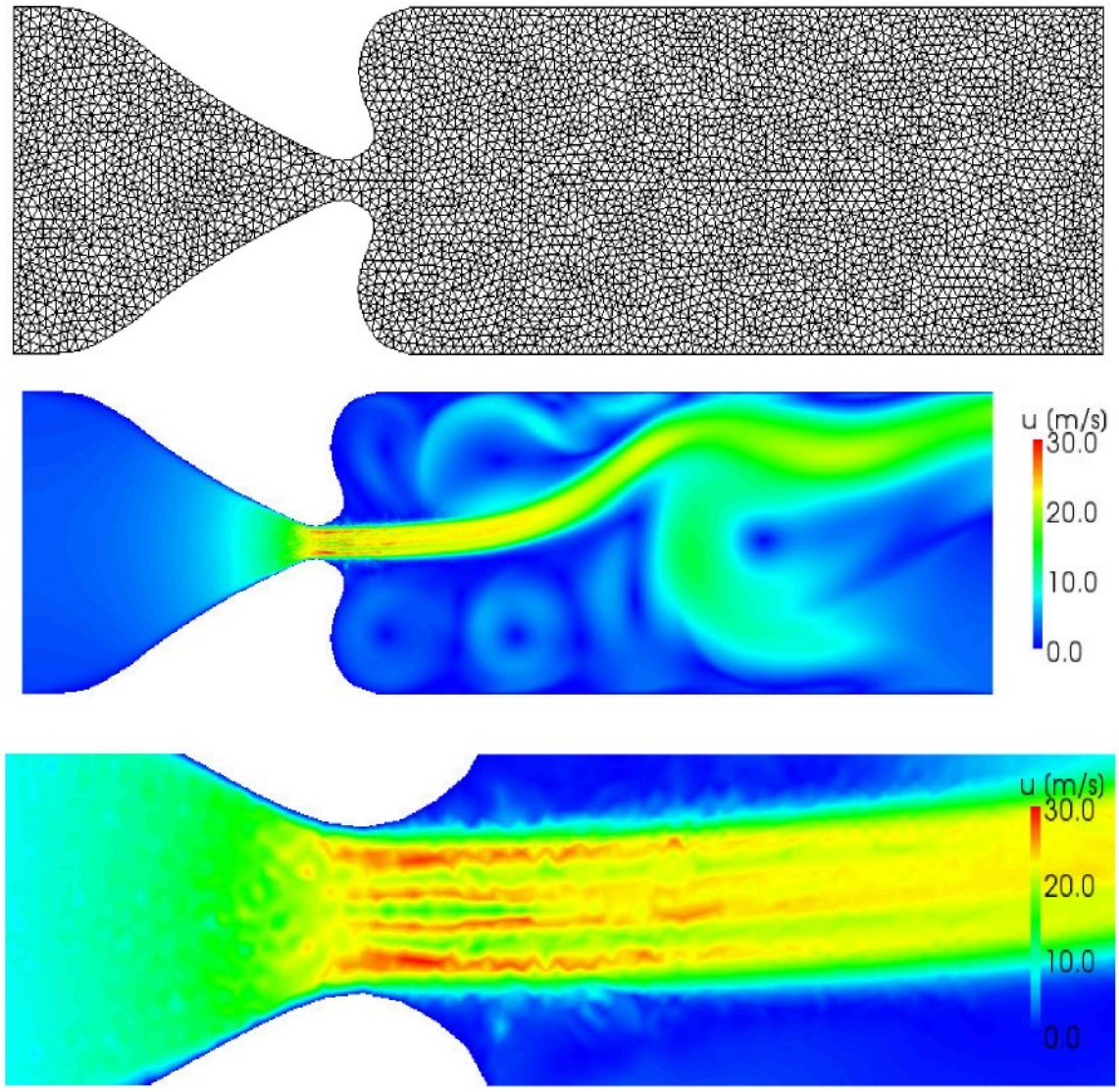
Jak již bylo několikrát v předchozích kapitolách zmíněno, sítě o velkém počtu neúměrně zatěžují výpočetní techniku. Pro demonstraci této skutečnosti byl na modelu hlasivky proveden výpočet rychlosti a tlaku. Výpočty provedl můj vedoucí Petr Šidlof a dodal i grafické výstupy rychlostního pole. Pro porovnání byli použity celkem tři sítě. První obsahuje 8091 elementů typu trojúhelník, druhá obsahuje 16055 elementů typu trojúhelník. Obě tyto sítě byly izotropní. Třetí síť pak byla lokálně zjemněná a obsahovala 11902 elementů typu trojúhelník.

Na sítí byla pro vektorové rychlostní pole před samotným výpočtem provedena kubická interpolace (z lineárních elementů byly vytvořeny elementy třetího řádu) , pro skalární tlakové pole kvadratická interpolace (elementy druhého řádu). Proto je počet řešených neznámých takto vysoký. Přehled důležitých hodnot při výpočtu rychlosti byl shrnut do tabulky (viz. tab. 5.1). Jak je v tabulce vidět, čas potřebný na výpočet jednoho časového kroku roste. Při dvojnásobném počtu elementů se čas na výpočet jednoho kroku ztrojnásobí. A náročnost na paměti počítače se zvýší na dvou a půl násobek.

Tab. 5.1: Přehled důležitých hodnot

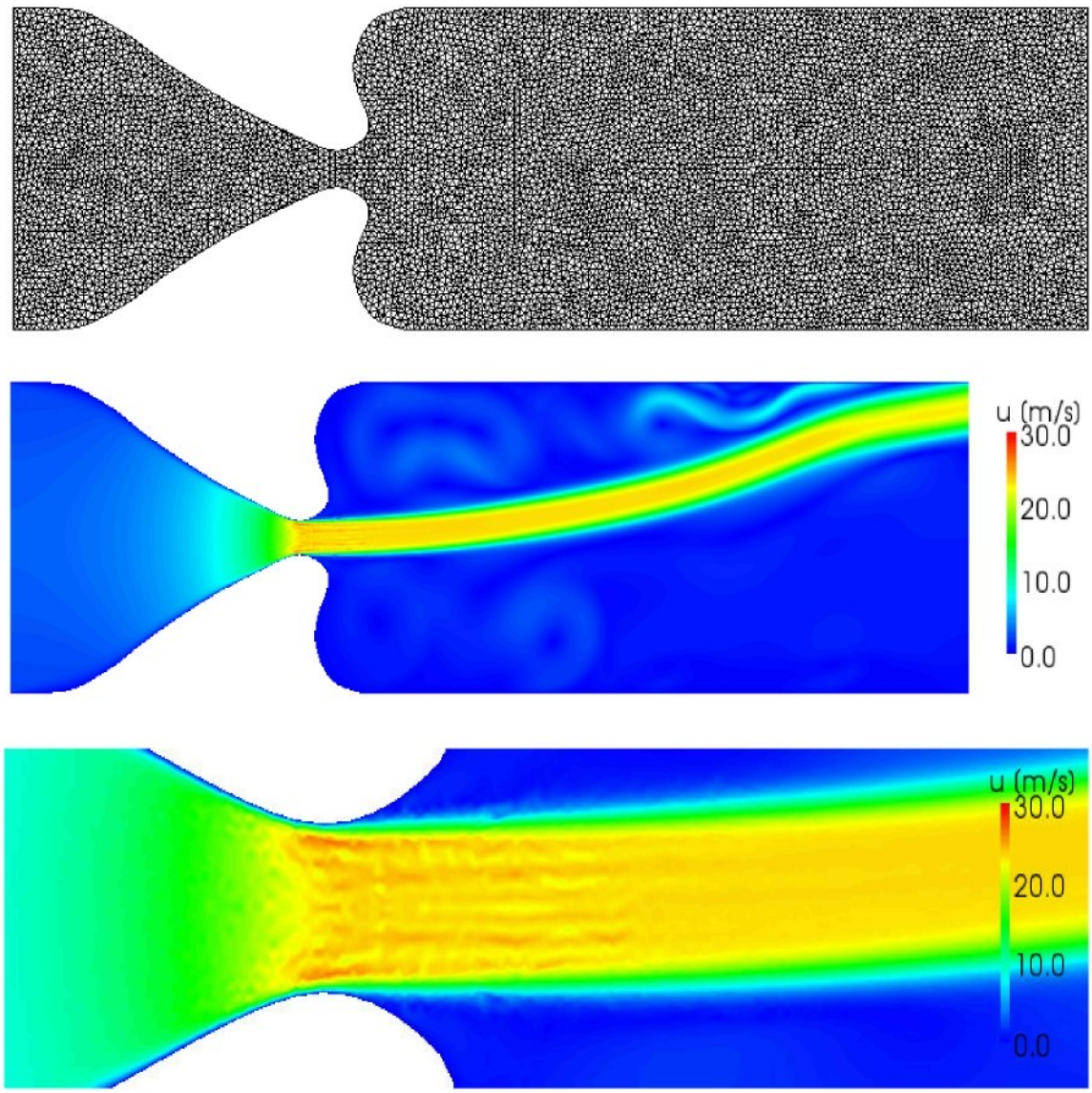
počet elementů v síti	čas na vypočtení jednoho časového kroku	Zatížení paměti počítače	počet řešených neznámých (rozměr matice)	počet nenulových prvků v matici [hodnota v milionech]
8091	33,5 s	250 MB	36500	14
11902	62 s	425 MB	54000	25
16055	95 s	625 MB	74000	35

Řešení na izotropní síti která osahuje 8091 elementů je patrné, že síť je příliš hrubá a řešení je nesprávné v oblasti zúžení vznikají oscilace což je nepřípustné (viz. obr. 5.3).



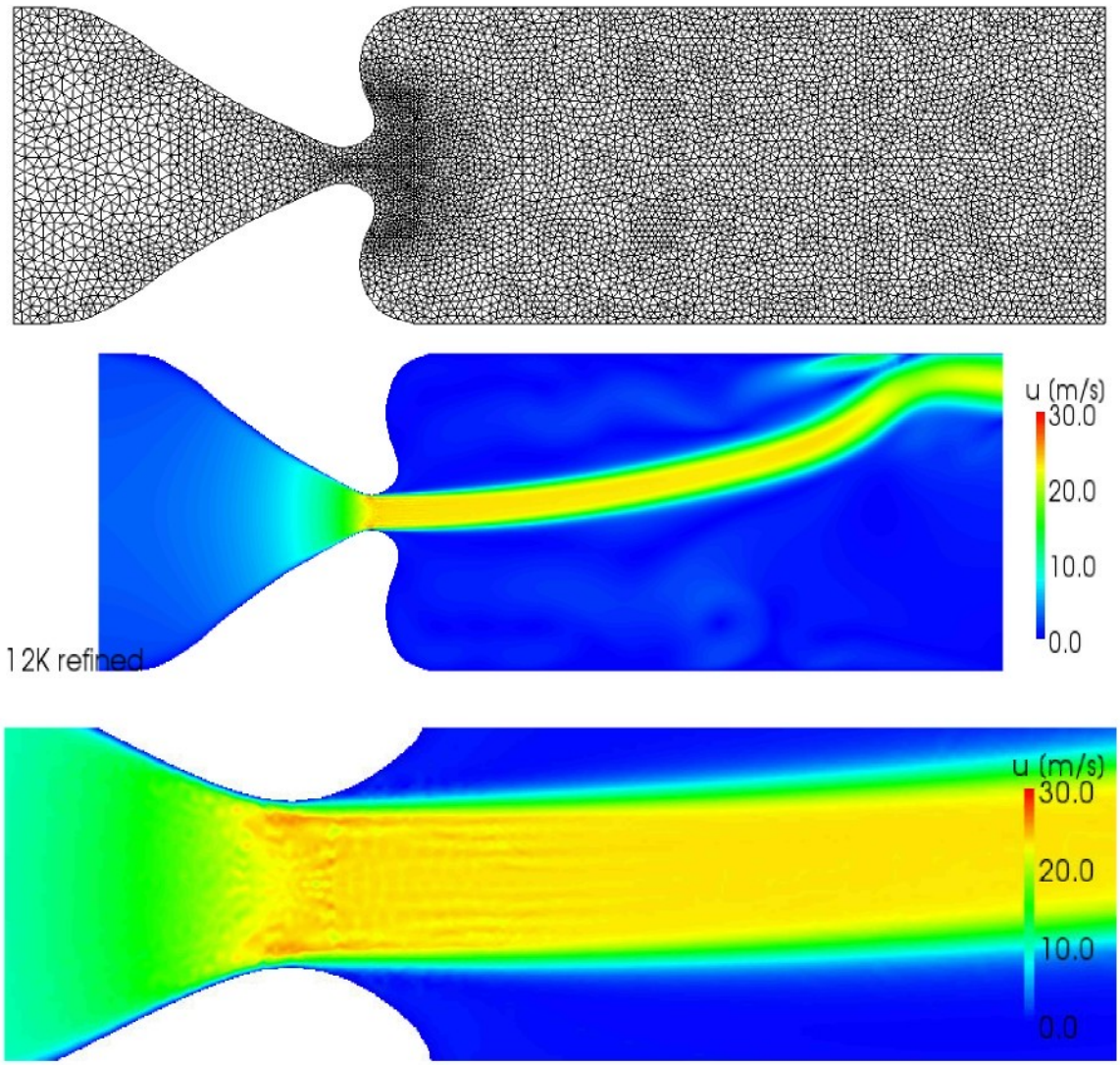
Obr. 5.3: Diskretizační síť společně s výsledným rychlostním polem 8091 elementů

Rychlostní pole řešené na izotropní síti, která obsahuje 16055 elementů je o poznání lepší přesto v oblasti zúžení dochází k určité nepřesnosti (viz. obr. 5.4).



Obr. 5.4: Diskretizační sít společně s výsledným rychlostním polem 16055 elementů

Rychlostní pole naopak u lokálně zjemněné sítě dosahuje nejlepšího výsledku, vzniklé oscilace jsou nejmenší. V teto síti byla také velikost elementu v zúžení nejmenší (viz. obr. 5.5), přestože počet elementů byl pouhých 11902.



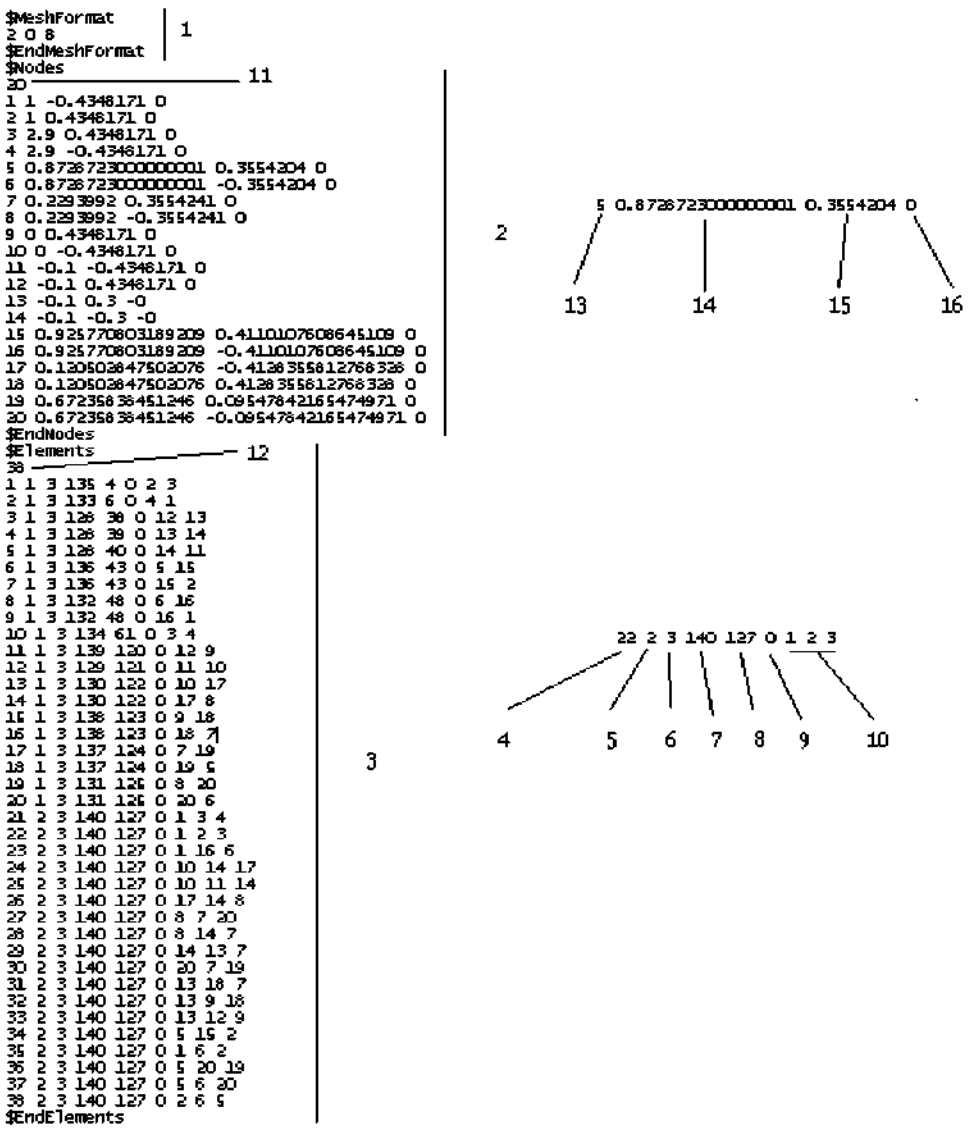
Obr. 5.5: Diskretizační sít společně s výsledným rychlostním polem 11902 elementů

6 Konvertor

6.1 Úvod

Tato kapitola popisuje druhou část bakalářské práce. Jedná se o program, který převádí výstupní data z generátoru sítě GMSH, ve formátu *.msh na soubor ve formátu *.maill, soubor potřebný pro výpočet. Oba soubory obsahují informace o diskretizační síti. Podrobná struktura obou formátů souboru bude popsána v následujících kapitolách stejně jako samotný konvertor. Soubor formátu *.maill je nutný pro výpočetní program.

6.2 Soubor *.msh a *.maill

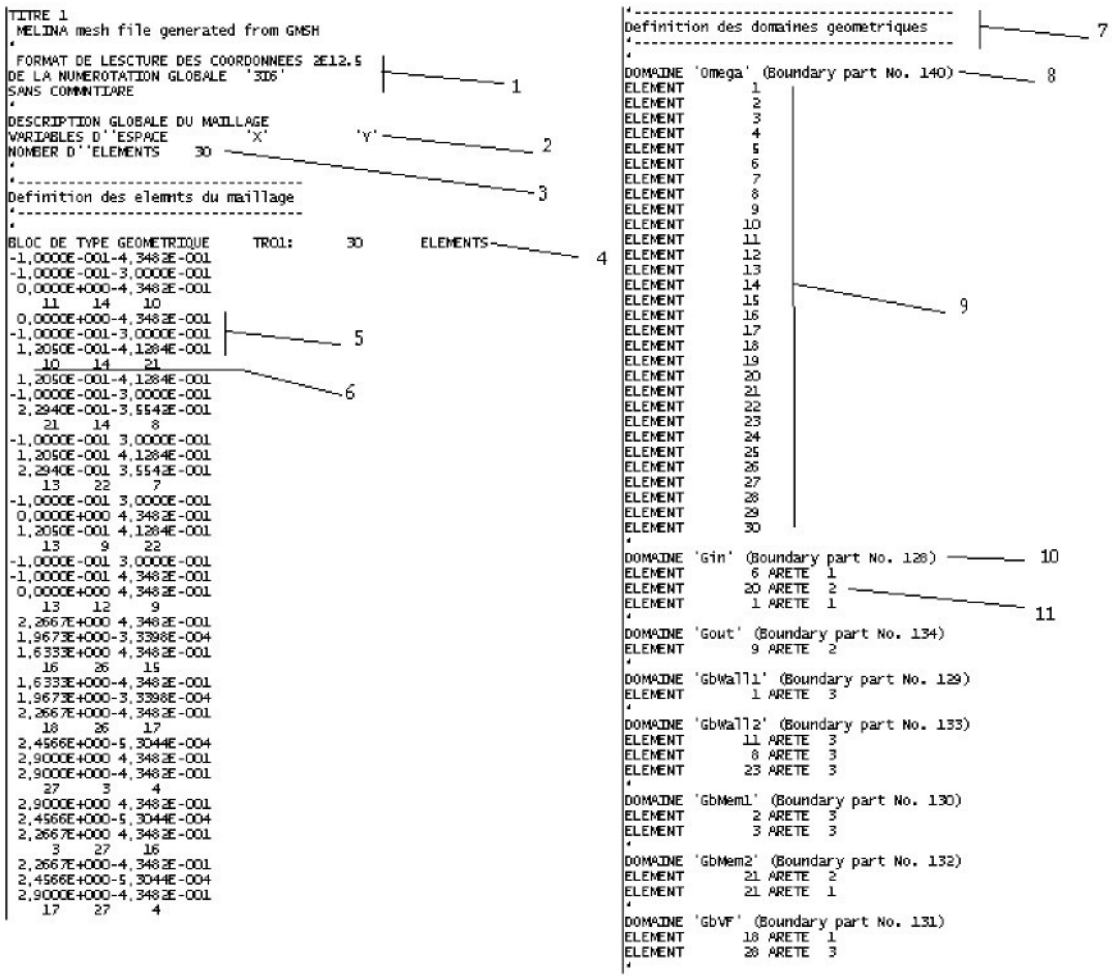


Obr. 6.1: Ukázka soboru ve formátu *.msh

Popis jednotlivých částí souboru ve formátu *.msh (viz. obr. 6.1) :

1. hlavička: obsahuje informaci o formátu souboru
2. uzly: tato pasáž začíná slovem $Nnodes končí slovem $EndNodes
3. elementy: tato pasáž začíná slovem $Elements končí slovem $EndElements
4. pořadové číslo elementu
5. typ elementu, seznam hodnot s přiřazeným tvaru elementu je napsán v referenčním manuálu.
6. počet programovatelných tagů (možnost nastavení požadovaných vlastností)
7. v modelu hlasivky určuje oblast (každá oblast pro okrajové podmínky má své číslo)
8. číslo čáry nebo plochy kde se element nachází (každá čára i oblast má přiřazené číslo)
9. volný tag
10. pořadová čísla uzlů, které určují element (počet se liší dle tvaru)
11. celkový počet uzlů
12. celkový počet elementů
13. pořadové číslo uzlu
14, 15, 16 souřadnice uzlu (x, y, z)

Tento soubor obsahuje kompletní informace o síti. Pořadová čísla jsou od hodnoty 1 a zvětšují se u každého dalšího elementu o jedničku (přirozená čísla). Podobný popis formátu souborů *.msh je v referenčním manuálu str. 67 až str. 72 .[2]



Obr.6.2: Ukázka souboru ve formátu *.maill

Popis jednotlivých částí souboru ve formátu *.maill (viz. Obr 6.2)

1. hlavička souboru, ve které je určen formát reálných čísel a čísel typu integer, pro zadávání souřadnic a indexů uzlů.
2. určení souřadného systému
3. udává celkový počet všech typů elementů
4. určí typy jednotlivých elementů a jejich počet
5. *x* a *y* souřadnice uzlů
6. indexy uzlu jak jdou po postupně na elementu
7. hlavička, která určuje část souboru pro zadávání okrajových podmínek
8. uvedení řešené oblasti, obsah závorky slouží jako komentář
9. seznam všech elementů které se nacházejí v řešené oblasti
10. identifikátor oblasti
11. seznam elementů a jejich hran, na které má být zadána okrajová podmínka

6.3 Program na zpracování souborů

Pro vytvoření konvertoru bylo vybráno programovací prostředí Lazarus IDE. Jedná se o volně šiřitelný program, který je dostupný na stránkách vývojářů. Lze nainstalovat na všechny počítače s operačními systémy Windows, Linux a Max OS. Pracuje na free pascal compileru. Je tedy velice podobný programovacímu jazyku delphi, se kterým mám zkušenosti.

Samotný program je napsán jako konzolová aplikace, kterou je možné spustit jak pod operačním systémem Windows, tak pod Linux. Program je funkční pro soubory obsahující 2D sítě. Program je plně univerzální pro jakoukoliv trojúhelníkovou síť vygenerovanou v programu GMSH a převede jí na soubor ve formátu *.maill . Program je psán přehledně pomocí procedur, které se postupně volají. Procedur je 8, dále program obsahuje část, ve které jsou zapsány veškeré konstanty, jako jsou rozměry statických polí, texty pro hlavičku, atd. Umístění těchto důležitých parametrů do konstant je z důvodu přehlednosti. Programu je po spuštění zadán z příkazové řádky název souboru ve formátu *.msh a *.geo, název souboru je brán jako parametr. Soubor ve formátu *.geo nese informace o počtu oblastí, plus přiřazené identifikační číslo. Při nalezení souboru ve formátu *.msh vytvoří soubor ve formátu *maill, který má stejné jméno a začne provádět jednotlivé procedury. Jednotlivé procedury jsou uvedeny v následujících odstavcích se stručným popisem co v programu dělají. Po skončení práce všechny použité soubory.

První procedura (nacteni_Nodu). V této proceduře je ze souboru ve formátu *.msh načtena nejprve hlavička, protože není pro konvertor důležitá nikam se neukládá. Poté následuje načítání jednotlivých souřadnic uzlů (x, y, z) do statických polí. Identifikační číslo uzlu odpovídá indexu v poli.

Druhá procedura (nacteni_Elementu). V této proceduře se ze souboru ve formátu *.msh načítá část, která obsahuje elementy. Nejprve se načte celkový počet elementů to slouží pro určení počtu opakování ve *for cyklu*. Všechny důležité informace jsou načteny opět do statických polích.

Třetí procedura (vytvorit_Hlavicku). Tato procedura vytvoří hlavičku pro soubor ve formátu *.maill. Text v hlavičce je uložen do konstant pokud by došlo ke změně této hlavičky, aby bylo možné jednotlivé věci jednoduše změnit.

Čtvrtá procedura (vypis_Elementu) vypíše upravené elementy do souboru ve formátu *maill. Pátá procedura (hlavicka_oblasti) má za úkol zapsat hlavičku oddělující část s elementy a část pro okrajové podmínky do souboru ve formátu *.maill.

Šestá procedura (nacteni_oblasti) načítá jednotlivé řádky ze souboru ve formátu *.geo . Ten obsahuje název oblasti pro okrajové podmínky a číselnou hodnotu, která odpovídá hodnotě, kterou u sebe nese element jako informaci pro okrajové podmínky. Soubor ve formátu *.geo bude popsán v kapitole 7 .

Sedmá procedura (rozdeleni_dle_oblasti) pracuje se statickými poli, ve kterých vyhledává důležité informace o jednotlivých elementech a podle nalezených informací rozdělí elementy do skupin. Počet skupin odpovídá počtu oblastí. Do jednotlivých skupin je pak uloženo číslo elementu a hrana, na kterou se okrajová podmínka bude zadávat. Zde bylo použito pole dynamické a to z důvodu že počet oblastí může být velký a mi ho před kompilací programu nevíme.

Osmá procedura (vypis_Oblasti) vypíše jednotlivé oblasti včetně svojí hlavičky a seznam elementů a hran, které příslušné oblasti náleží.

7 Generátor sítí GMSH

7.1 Popis programu

Tato kapitola bude věnována programu pro generování konečněprvkových sítí. Byl vybrán program GMSH jedná se o free software, což znamená že je volně šiřitelný a za jeho používání se nemusí platit. Licenční práva jsou přístupná na internetových stránkách toho softwaru *http://geuz.org/gmsh*. Jelikož tato práce má sloužit i jako stručný manuál k tomuto programu bude zde věnováno několik kapitol základní obsluze tohoto programu. Bližší informace lze pak naleznout v referenčním manuálu na stránkách o tomto programu. Dalším zdrojem informací pak může být tzv. Mailing Lists, kde se uživatel může zaregistrovat a psát své dotazy, které budou pomocí emailů rozeslaný všem ostatním uživatelům včetně vývojářům, kteří vám poskytnou cenné rady. Program byl vyvinut Christophe Geuzaine (Univerzita Lutych) a Jean-François Remacle (Katolická univerzita v Lovani).

Program GMSH je tří-dimenzionální generátor konečněprvkových sítí s vestavěným preprocessingem a postprocessingem postaveného na CAD engine. Jedná se o jednoduchý generátor sítí pro akademické problémy. GMSH je složen ze 4 modulů geometrie, mesh, solver a postprocessingem. Moduly a solver a postprocessingem v této práci popisovat nebudu, jelikož nebyly obsahem této práce. Ve všech modulech je možné zadávat instrukce z grafického prostředí nebo pomocí textových souborů. Jelikož GMSH osahuje vlastní skriptovací jazyk. Obě tyto možnosti jsou společně provázané změna v jednom prostředí vyvolá změnu v druhém prostředí. [2]

7.2 Výhody a nevýhody programu GMSH

Program GMSH je co se týče velikosti instalačního souboru velice malý. Prvotně byl vytvořen pro řešení akademických problémů, ale postoupen času se rozšířil i mimo tuto oblast. Bohužel díky tomu, že byl určen pro akademické prostředí a vývojářský tým je malý, má tento program i několik nevýhod. Výhody a nevýhody budou shrnuty v následujícím přehledu. [2]

Výhody:

- snadné a rychlé vytvoření geometrie díky zabudovaným funkcím
- malá hardwarová náročnost
- vkládání složitějších geometrií ve formátech jako je STEP a IGES
- vytváření 1D, 2D a 3D konečně prvkových sítí se standardními elementy
- vytváření animací
- každý modul nabízí několik možností výstupních formátů

Nevýhody:

- tvorba velkých konečně prvkových sítí trvá velice dlouho
- omezený počet elementů v řádu milionů elementů
- všechny konečně prvkové sítě jsou nestrukturované
- velmi těžké a časové náročné vytvořit složitou geometrii

7.3 Modul geometrie

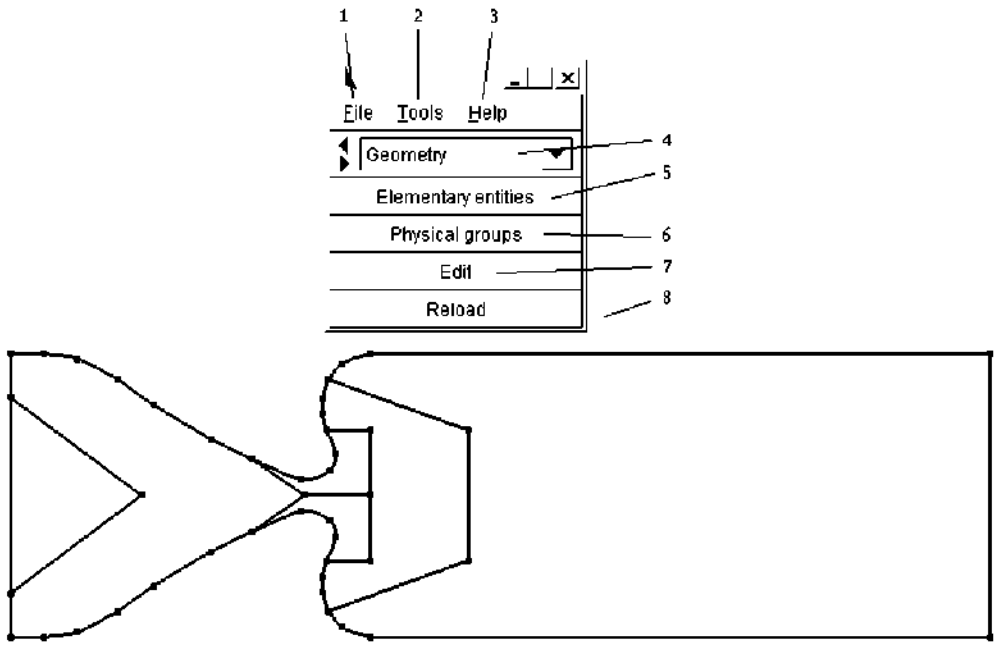
Modul geometrie je součástí programu, pro zadávání lze využít grafické rozhraní i textový soubor. Při konstrukci geometrie se postupuje následovně. Nejprve je nutné definovat body, dále tyto nadefinované body spojit přímkou, kružnicí a nebo obecnou křivkou. Poté vytvoří plochy, které jsou definovány pomocí spojnic bodů a nakonec je možné i vytvořit objemové (3D) geometrie. Plochy a objemy je nutné definovat pro pozdější zasíťování. Všechny prvky musí mít přiřazené své identifikační číslo. Je možné také vytvořit tzv. fyzické skupiny. To ho se využívá pak v modulu síť pro určení hraničních elementů pro zadávání okrajových podmínek.

Bod je v tomto modulu definován třemi souřadnicemi X, Y, Z a jeho identifikačním číslem a také obsahuje informaci o charakteristické délce. Charakteristická délka určuje maximální velikost elementu sítě v okolí tohoto bodu. Všechny definice bodu mohou být zadávány přímo jako hodnoty (v grafickém rozhraní) nebo parametricky či pomocí funkce (textový soubor).

Dalším prvkem geometrie jsou čáry, je jich několik druhů. Nejzákladnější je přímka. Přímka je definována pouze dvěma body a opět má přiřazené své specifické číslo. Definice pak vypadá následovně *LINES(identifikační číslo) = {identifikační číslo bodu1, identifikační číslo bodu2}*. Dalším druhem čáry je pak kružnice, ta je definována třemi body a opět svým charakteristickým číslem. Definice pak vypadá následně

Circle(specifické číslo) = {identifikační číslo počátečního bodu oblouku, identifikační číslo středu kruhu, identifikační číslo koncového bodu oblouku}. Nejobecnější je pak spline, jedná se o křivku, která je definována *n* počtem bodu, kde *n* je přirozené číslo. Definice pak je následující *Spline(identifikační číslo) = (seznam všech identifikačních čísel bodů, které leží na křivce).*

Definice ploch a objemů je pak podobná, opět mají své identifikační číslo. U plochy je pak uveden seznam jednotlivých čar, které tvoří uzavřenou smyčku, ve smyčce se pak dodržuje definovaný směr, proto mají některá identifikační čísla čar zápornou hodnotu. Objem je pak definován pomocí ploch. [2]



Obr. 7.1: Grafické rozhraní programu GMSH s otevřeným modulem geometrie

Popis modulu geometrie (viz obr. 7.1)

1. záložka File slouží k ukládání, otvírání a vytváření nových geometrii
2. záložka tools obsahuje prostředky jako je nastavení a další potřebné informace
3. možnost rychle se dostat do referenčního manuálu
4. pole pro výběr modulu
5. po rozkliknutí umožňuje zadávat, ale i mazat jednotlivé prvky geometrie
6. vytváření fyzických skupin
7. přístup do textového souboru a možnost ho měnit
8. vyvolání změn s souboru do grafického rozhraní

Výslédná geometrie je uložena do souboru ve formátu *.geo, náhled do souboru (viz obr. 7.2). V našem případě je charakteristická délka zadána pomocí parametru pro snažší změnu při zjemňování sítě. U fyzikálních entit není identifikační číslo ale jméno. Jméno je proměna do které je v souboru *Oblasti parametrizovaný tvar přiřazena hodnota* přiřazena hodnota. Opět to slouží pro snadné měnění názvů. Dále to pomáhá při určování, který element kde leží při zadávání okrajových podmínek jelikož soubor *Oblasti parametrizovaný tvar přiřazena hodnota* je využíván i v konvertoru. [2]

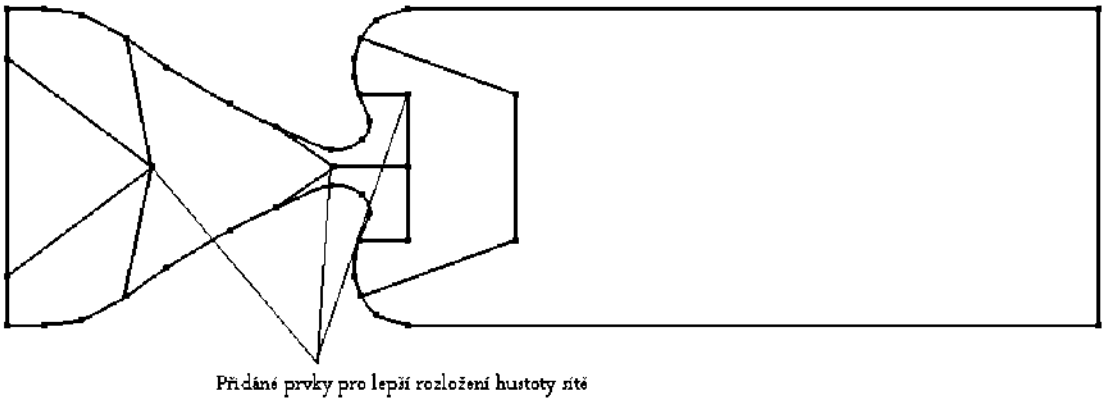
```
lc1 = 0.015;
lc2 = 0.005;
lc3 = 0.05;
Point(1) = {1, -0.4348171, 0, lc1};
Point(2) = {1, 0.4348171, 0, lc1};
Point(3) = {2.9, 0.4348171, 0, lc1};
Point(4) = {2.9, -0.4348171, 0, lc1};
Point(5) = {0.9150168, 0.4053319, 0, lc1};
Point(6) = {0.9150168, -0.4053319, 0, lc1};
Point(7) = {0.8728723000000001, 0.3554204, 0, lc1};
Point(8) = {0.8728723000000001, -0.3554204, 0, lc1};
Point(9) = {0.8548134, -0.2992305, 0, lc1};
Point(10) = {0.8548134, 0.2992305, 0, lc1};
Point(11) = {0.8539543000000001, 0.249552, 0,lc1};
Point(12) = {0.8539543000000001, -0.249552, 0, lc1};
Point(13) = {0.8668663, 0.2000227, 0, lc2};
Point(14) = {0.8668663, -0.2000227, 0, lc2};
Point(15) = {0.8943691, 0.1278864, 0, lc2};
Point(16) = {0.8943691, -0.1278864, 0, lc2};
Point(17) = {0.8766139, 0.07710425, 0, lc2};
Point(18) = {0.8766139, -0.07710425, 0, lc2};
Point(19) = {0.7915073, 0.04907636, 0, lc2};
Point(20) = {0.7915073, -0.04907636, 0, lc2};
Point(21) = {0.6371318, 0.111797, 0, lc1};
Point(22) = {0.6371318, -0.111797, 0, lc1};
Point(23) = {0.5135537, 0.1723353, 0, lc1};
Point(24) = {0.5135537, -0.1723353, 0, lc1};
Point(25) = {0.3383989, -0.2771944, 0, lc1};
Point(26) = {0.3383989, 0.2771944, 0, lc1};
Point(27) = {0.2293992, 0.3554241, 0, lc1};
Point(28) = {0.2293992, -0.3554241, 0, lc1};
Point(29) = {0.1046835, -0.41900402, 0, lc1};
Point(30) = {0.1046835, 0.41900402, 0, lc1};
Point(31) = {0, 0.4348171, 0, lc1};
Point(32) = {0, -0.4348171, 0, lc1};
Point(33) = {-0.1, -0.4348171, 0, lc1};
Point(34) = {-0.1, 0.4348171, 0, lc1};
Point(35) = {-0.1, 0.3, -0, lc3};
Point(36) = {-0.1, -0.3, -0, lc3};
Point(37) = {0.3, -0, -0, lc3};
Point(38) = {1, -0, -0, lc2};
Point(39) = {1, -0.2, -0, lc2};
Point(40) = {1.3, 0.2, -0,lc1};
Point(41) = {1.3, -0.2, 0, lc1};
Point(42) = {1, 0.2, -0, lc2};
Point(43) = {0.8, -0, -0, lc2};
Line(1) = {34, 31};
Line(2) = {2, 3};
Line(3) = {3, 4};
Line(4) = {4, 1};
Line(5) = {32, 33};
Line(6) = {33, 36};
Line(7) = {36, 35};
Line(8) = {35, 34};
Spline(9) = {31, 30, 27};
Spline(10) = {7, 5, 2};
Spline(11) = {32, 29, 28};
Spline(12) = {8, 6, 1};
Line(13) = {35, 37};
Line(14) = {36, 37};
Line(15) = {43, 38};
Line(16) = {38, 42};
Line(17) = {42, 13};
Line(18) = {40, 7};
Line(19) = {40, 41};
Line(20) = {41, 8};
Point(35) = {-0.1, 0.3, -0, lc3};
Point(36) = {-0.1, -0.3, -0, lc3};
Point(37) = {0.3, -0, -0, lc3};
Point(38) = {1, -0, -0, lc2};
Point(39) = {1, -0.2, -0, lc2};
Point(40) = {1.3, 0.2, -0,lc1};
Point(41) = {1.3, -0.2, 0, lc1};
Point(42) = {1, 0.2, -0, lc2};
Point(43) = {0.8, -0, -0, lc2};
Line(1) = {34, 31};
Line(2) = {2, 3};
Line(3) = {3, 4};
Line(4) = {4, 1};
Line(5) = {32, 33};
Line(6) = {33, 36};
Line(7) = {36, 35};
Line(8) = {35, 34};
Spline(9) = {31, 30, 27};
Spline(10) = {7, 5, 2};
Spline(11) = {32, 29, 28};
Spline(12) = {8, 6, 1};
Line(13) = {35, 37};
Line(14) = {36, 37};
Line(15) = {43, 38};
Line(16) = {38, 42};
Line(17) = {42, 13};
Line(18) = {40, 7};
Line(19) = {40, 41};
Line(20) = {41, 8};
Spline(25) = {27, 26, 23, 21};
Spline(26) = {21, 19, 17, 15, 13};
Spline(27) = {13, 11, 10, 7};
Spline(28) = {28, 25, 24, 22};
Spline(29) = {22, 20, 18, 16, 14};
Spline(30) = {14, 12, 9, 8};
Line Loop(31) = {7, 13, -14};
Plane Surface(32) = {31};
Line Loop(33) = {9, 25, -24, 23, -28, -11, 5, 6, 14, -13, 8, 1};
Plane Surface(34) = {33};
Line Loop(35) = {26, -17, -16, -15, 24};
Plane Surface(36) = {35};
Line Loop(37) = {21, 22, -29, -23, 15};
Plane Surface(38) = {37};
Line Loop(39) = {27, -18, 19, 20, -30, -22, -21, 16, 17};
Plane Surface(40) = {39};
Line Loop(41) = {2, 3, 4, -12, -20, -19, 18, 10};
Plane Surface(42) = {41};
Include "Oblasti_parametrizovany_tvar.geo";
Physical Surface(Omega) = {42, 40, 36, 38, 34, 32};
Physical Line(Gin) = {8, 7, 6};
Physical Line(Gout) = {3};
Physical Line(GbWall1) = {5};
Physical Line(GbWall2) = {4};
Physical Line(GbMem1) = {11};
Physical Line(GbMem2) = {12};
Physical Line(GbVF) = {28, 29, 30};
Physical Line(GuWall1) = {1};
Physical Line(GuWall2) = {2};
Physical Line(GuMem1) = {9};
Physical Line(GuMem2) = {10};
Physical Line(GuVF) = {25, 26, 27};
```

Obr. 7.2: Soubor ve formátu *.geo

7.4 Modul mesh

Modul mesh je opět součástí programu GMSH a navazuje přímo na předchozí modul. Veškeré nastavení parametrů sítě se již totiž provádí při zadávání geometrie nebo z grafického rozhraní v modulu mesh. Tento generátor nabízí celou řadu elementů včetně jejich vyšších řádů, nejvyšší řád je 5. Tuto řadu tvoří úsečka, trojúhelník, čtyřúhelník, čtyřstěn, šestistěn, kvádr a pyramida.

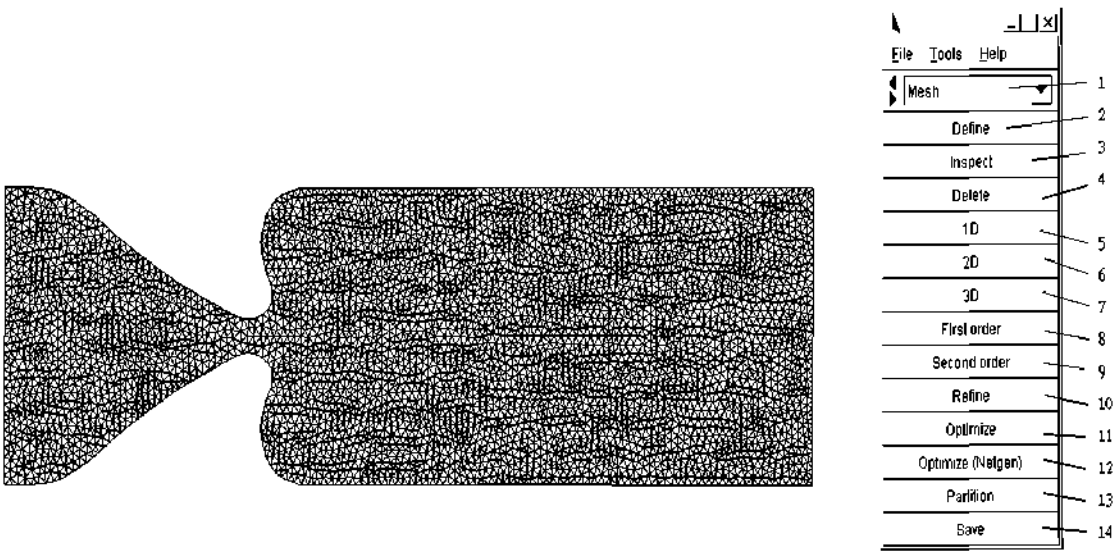
Všechny sítě jsou považovány za nestrukturované. Nestrukturované algoritmy pro 2D jsou MeshAdapt, Delaunayova a Frontal. Pro 3D je to pak Tetgen+Delaunay a Netgen.



Obr. 7.3: Základní geometrie s přidanými prvky pro lepší rozložení hustoty sítě

Velikost elementů lze ovlivnit několika způsoby. Jedním z nich je nastavení charakteristické délky při zadávaní bodu geometrie. Charakteristická délka určuje jakou maximální délku může mít hrana elementu. Pro lepší rozdělení hustoty sítě je občas nutné zadat navíc prvky geometrie(body, přímky). Toho bylo využito i v této práci, kde byly přidány prvky v okolí zúžení hlasivek a na výstupu. Ukázka mnou zvoleného rozložení přídavných prvku (viz. obr. 7.3). Algoritmy pak přechody mezi oblastmi s různou hustotou vyřeší velmi dobře bez toho aniž by vznikaly nevhodné elementy. Velikost elementu se plynule od zadaného bodu mění(zmenšuje nebo zvětšuje)

Dalším způsobem, jak ovlivnit velikost elementu sítě v určité oblasti, je pomocí pole. Pole umožňuje několik nastavení, podle kterých se velikost elementů bude řídit. Lze to pomocí matematické funkce, v závislosti na vzdálenosti od geometrických prvků, atd. . Další možností změny hustoty sítě je pak v grafickém rozhraní pomocí funkce *REFINE*. Síť je pak uložena do souboru ve formátu **.msh*, popis jednotlivých částí tohoto souboru je uveden v kapitole 6.2. [2]



Obr. 7.4: Grafické rozhraní programu GMSH s otevřeným modulem Mesh

Popis modulu mesh (viz. obr. 7.4)

1. zvolen modul mesh
2. zde se dá nastavit charakteristická délka, překreslení sítě a výběr pole
3. zobrazí informace o elementu na, který se klikne
4. vymaže element, čáru, atd.
5. 1D síť
6. 2D síť
7. 3D síť
8. element prvního řádu (lineární)
9. element druhého řádu
10. zjemnění element, zmenší na polovinu
11. optimalizace pro zlepšení kvality elementu typu čtyřstěn
12. optimalizace pro zlepšení kvality elementu typu čtyřstěn
13. –
14. uložit

Závěr

Úkolem této práce bylo seznámení s problematikou generování sítí pro výpočty metodou konečných prvků. Poté vybrat z velkého množství dostupných generátorů konečně prvkových sítí jeden generátor. Otestovat jeho možnosti hlavně s důrazem na lokální zjemnění sítě. A nakonec vytvořit na zadané geometrii síť vhodnou pro výpočet metodou konečných prvků.

Před samotným vybráním vhodného softwaru jsem nastudoval potřebné informace k problematice generování sítí, hlavně pak sítí lokálně zjemněných. Po odzkoušení několika volně stažitelných programů byl vybrán program GMSH. Tento program umožňuje jednoduché generování lokálně zjemněných sítí a generování všech používaných typů elementu.

Po dokonalém seznámení s programem, bylo vytvořeno několik sítí s různými parametry. Parametry jsou počet elementů a hustota sítě v různých oblastech. Tyto sítě pak byly využity pro řešení úlohy proudění vzduch v lidských hlasivkách.

Pro porovnání kvality výpočtu byly využity sítě izotropní a lokálně zjemněná. Z vizualizace výsledků pak bylo patrné, že pro získání srovnatelného řešení bylo u lokálně zjemněné sítě potřeba o ¼ méně elementů. To vedlo k rychlejšímu řešení úlohy a k menšímu zatížení výpočetního hardwaru.

Před samotným výpočtem pak bylo nutné výstupní soubor ve formátu *.msh z programu GMSH převést na soubor ve formátu *.maill. Ten slouží jako zdroj informací o síti pro samotný výpočetní program. Program na převod těchto souborů je vytvořen tak, aby převáděl všechny soubory, které obsahují 2D trojúhelníkové sítě s elementy prvního řádu. Rozšíření tohoto programu bude povedeno jako součást mé další práce. Poté už by měl být program na převod souborů plně univerzální.

Použitá literatura

[1] Bathe, Klaus-Jürgen. *Finite Element Procedures.* Prentice Hall, Pearson Education, Inc., 2006.

[2] Gmsh Reference Manual The documentation for Gmsh 2.3
A finite element mesh generator with built-in pre- and post-processing facilities
29 March 2009

[3] W. Hackbusch, S.A. Sauter - Composite finite elements for problems containing small geometric details, Computing and Visualization in Science 1, 1997

[4] cviceni_02_MTP_2009.pdf - učební texty

[5] Introduction to Finite Element Method - T.H. Kwon 2005

[6] http://geuz.org/gmsh/gallery/piece3.gif

[7] http://en.wikipedia.org/wiki

[8] apm-prednasky_06-04-05.pdf - Milan Hokr 5. dubna 2006

Příloha

V příloze je vylistovaná zdrojový kód konvertoru souborů

```
program msh2maill;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  Classes, SysUtils, CustApp , DateUtils , Strutils
  { you can add units after this };


const HLAVICKA_RADEK1 = 'TITRE 1';
     HLAVICKA_RADEK2 = ' MELINA mesh file generated by GMSH/msh2maill ';
     HLAVICKA_RADEK3 = ' FORMAT DE LECTURE DES COORDONNEES ';
     PLATNYCH_CIFER = '5';
     POCET_ZNAKU_REAL = '12';
     FORMAT_2REAL = '2E' + POCET_ZNAKU_REAL + '.' + PLATNYCH_CIFER;
     HLAVICKA_RADEK4 = '       DE LA NUMEROTATION GLOBALE  ';
     FORMAT_3INTEGER = '3I6' ;
     FORMAT_MSH_INTEGER = '%6d';
     HLAVICKA_RADEK5 = '       SANS COMMENTAIRE';
     HLAVICKA_RADEK6 = 'DESCRIPTION GLOBALE DU MAILLAGE';
     HLAVICKA_RADEK7 =' VARIABLES D' + '''' + '''' + 'ESPACE  ' + '''' +
                'X' + '''' + '    ' + '''' + 'Y' + '''';
     HLAVICKA_RADEK8 = '  NOMBRE D' + '''' + '''' + 'ELEMENTS      ';
     HLAVICKA_RADEK9 = '* Definition des elements du maillage ' ;
     HLAVICKA_RADEK10  = 'BLOC DE TYPE GEOMETRIQUE ';
     HLAVICKA_RADEK10a = ' ELEMENTS';
     HLAVICKA_RADEK11  = '* Definition des domaines geometriques';
     HLAVICKA_RADEK12  ='FIN (du fichier de maillage)';
     HLAVICKA_KOMENTAR = '*';
     HLAVICKA_KOMENTAR_LINKA = HLAVICKA_KOMENTAR +   ' ----------------------------------';

     TYP_TROJUHELNIK = 'TR01';
//     TYP_CTYRUHELNIK = 'TR?????';

     MAX_POCET_ELEMENTU = 100000;
     MAX_POCET_NODU = 100000;
     MAX_POCET_OBLASTI = 100;
     MAX_VRCHOLU_NA_ELEMENT= 4;
     ADRESAR_VYSTUP= '.'+PathDelim;
     INC_DYNAMIC_ARRAY = 1000;


Type
{Konvertor}
  T_Souradnice = array [1..MAX_POCET_NODU ] of real;
  T_Vlastnosti_elementu = array [1..MAX_POCET_ELEMENTU] of longint;
  T_Nod = array[1..MAX_POCET_NODU,1..MAX_VRCHOLU_NA_ELEMENT]of longint;
  T_element= array[1..MAX_POCET_ELEMENTU,1..MAX_VRCHOLU_NA_ELEMENT]   of longint;
```

```
  T_oblasti=array[1..MAX_POCET_OBLASTI] of record
                                                    jmeno:string;
                                                    cislo:longint;
                                                  end;
TKonvertor = class(TCustomApplication)

protected
  procedure DoRun; override;

public
    constructor Create(TheOwner: TComponent); override;
    destructor Destroy; override;
    procedure WriteHelp; virtual;

private
    procedure vytvorit_Hlavicku(var f_maill : textfile);
    procedure nacteni_Nodu(var f_msh : textfile );
    procedure nacteni_Elementu(var f_msh : textfile );
    procedure vypis_Elementu(var f_maill : textfile );
    procedure hlavicka_oblasti(var f_maill :textfile);
    procedure rozdeleni_dle_oblasti();
    procedure konverze(var jmeno_soubor_msh : TFileName);
    procedure nacteni_oblasti(var f_geo: textfile);
    procedure vypis_oblasti(var f_maill:textfile);

    jmeno,radek: string;
    file_msh, file_maill,file_geo: textfile;
    jmeno_soubor_msh,jmeno_maill:TFileName;
    pocet_elementu,pocet_nodu,pocet_elementu1,pocet_oblasti,cislo :longint;
    element_TR01, element_ctverec,element_primka,cislo_elementu,cislo_hrany:longint;
    Souradnice_x,Souradnice_y,Souradnice_z:T_Souradnice;
    Typ,Prislusna_oblast:T_Vlastnosti_elementu;
    element_trojuhelnik,element_ctyruhelnik:T_element;
    Nod:T_Nod;
    oblasti:T_oblasti;
    seznam_oblasti:array of array of string; // posledni cast *.maill - soupis obl.
    N_polozek_oblasti:array of integer; //pocet polozek (radku) u jednotlivych oblasti
    jmeno_soubor_domains:string;

end;
procedure TKonvertor.vytvorit_Hlavicku(var f_maill : textfile);
begin
writeln(f_maill , HLAVICKA_RADEK1);
writeln(f_maill , HLAVICKA_RADEK2);
writeln(f_maill , HLAVICKA_KOMENTAR);
writeln(f_maill , HLAVICKA_RADEK3,"",FORMAT_2REAL,"");
writeln(f_maill , HLAVICKA_RADEK4,"", FORMAT_3INTEGER,"");
writeln(f_maill , HLAVICKA_RADEK5);
writeln(f_maill , HLAVICKA_KOMENTAR);
writeln(f_maill , HLAVICKA_RADEK6);
writeln(f_maill , HLAVICKA_RADEK7);
writeln(f_maill , HLAVICKA_RADEK8, pocet_elementu1);
writeln(f_maill , HLAVICKA_KOMENTAR);
writeln(f_maill , HLAVICKA_KOMENTAR_LINKA);
writeln(f_maill , HLAVICKA_RADEK9);
writeln(f_maill , HLAVICKA_KOMENTAR_LINKA);
writeln(f_maill , HLAVICKA_KOMENTAR);
writeln(f_maill , HLAVICKA_RADEK10, TYP_TROJUHELNIK, ':      ', element_TR01,
          HLAVICKA_RADEK10a);
end;
```

```
procedure TKonvertor.nacteni_Nodu(var f_msh : textfile );
var
i:longint;
cislo_nodu: longint;
begin
for i:=1 to 4 do readln(f_msh);
 readln(f_msh, pocet_nodu);

 for i:=1 to pocet_nodu do
    begin
      readln(f_msh,cislo_nodu,Souradnice_x[i],Souradnice_y[i],Souradnice_z[i]);
    end;
end;

Procedure Tkonvertor.nacteni_Elementu(var f_msh : textfile);
var
i,j,k,l,pomocna,pocet_tagu,hranaobjektu,cislo_elemntu:longint;
begin
k:=1;
l:=1;
for i:=1 to 2 do readln(f_msh);
readln(f_msh, pocet_elementu);

for i:=1 to pocet_elementu do
  begin
  read(f_msh , cislo_elemntu,Typ[i], pocet_tagu);

  case pocet_tagu of
  0:  begin
      end;
  1:  begin
       read(f_msh , Prislusna_oblast[i]);
      end;
  2:  begin
       read(f_msh , Prislusna_oblast[i] , hranaobjektu  );
      end;
  3:  begin
       read(f_msh ,Prislusna_oblast[i], hranaobjektu , pomocna );
      end;
  end;

case Typ[i] of
   1:
    begin // Line element
     for j:=1 to 2 do
      begin
       read(f_msh,Nod[i,j]);
       end;
     element_primka:= element_primka +1;
    end;
   2:
    begin // Triangle
     for j:=1 to 3 do
      begin
       read(f_msh,Nod[i,j]);
       element_trojuhelnik[k,j]:=Nod[i,j];
      end;
     inc(k);
     inc(element_TR01);
    end ;
```

```
   3:
    begin // Quadrangle
     for j:=1 to 4 do
      begin
       read(f_msh,Nod[i,j]):
       element_ ctyruhelnik[l,j]:=Nod[i,j]:
      end;
      inc(element_ctverec):
      inc(l);
     end;
   end;
   end;
   pocet_elementu1:= element_TR01+element_ctverec:
end;


Procedure Tkonvertor.rozdeleni_dle_oblasti();
var
i,j,k,l,index_oblasti,len,ix:longint:
begin
SetLength(N_polozek_oblasti,MAX_POCET_OBLASTI);
for i:=1 to MAX_POCET_OBLASTI do N_polozek_oblasti[i]:=0;
SetLength(seznam_oblasti,MAX_POCET_OBLASTI):

for i:=1 to pocet_elementu do
begin
   j:=0;
   repeat
    inc(j);
    If j > MAX_POCET_OBLASTI then begin
     writeln('Domain number ',Prislusna_oblast[i],' does not exist');
     readln;
     Halt;
    end;
   until oblasti[j].cislo = Prislusna_oblast[i];
   index_oblasti:=j; //index oblasti, ktery prislusi element i
   N_polozek_oblasti[index_oblasti]:=N_polozek_oblasti[index_oblasti]+1:
   ix:=N_polozek_oblasti[index_oblasti];

   len:=Length(seznam_oblasti[index_oblasti]); //pokud nestaci dylka pole, zvetsime o inkrement
   if len <= N_polozek_oblasti[index_oblasti] then begin;
    SetLength(seznam_oblasti[index_oblasti],len+INC_DYNAMIC_ARRAY);
    writeln('Resizing dynamic arrays [',index_oblasti,'] : ',
     len+INC_DYNAMIC_ARRAY, 'entries');
   end;

   case Typ[i] of
     1:
   begin
for k:=1 to element_TR01 do
 begin
  if( ((Nod[i,1]=element_trojuhelnik[k,1])or(Nod[i,1]=element_trojuhelnik[k,2])
  or(Nod[i,1]=element_trojuhelnik[k,3]))and((Nod[i,2]=element_trojuhelnik[k,1])
  or (Nod[i,2]=element_trojuhelnik[k,2]) or (Nod[i,2]=element_trojuhelnik[k,3])))
    then cislo_elementu:=k;
```

```
if ((Nod[i,1] = element_trojuhelnik[k,1])and
            (Nod[i,2] = element_trojuhelnik[k,2])) then cislo_hrany:=1;
  if ((Nod[i,1] = element_trojuhelnik[k,2])and
            (Nod[i,2] = element_trojuhelnik[k,3])) then cislo_hrany:=2;
  if ((Nod[i,1] = element_trojuhelnik[k,3])and
            (Nod[i,2] = element_trojuhelnik[k,1])) then cislo_hrany:=3;
  if ((Nod[i,2] = element_trojuhelnik[k,1])and
            (Nod[i,1] = element_trojuhelnik[k,2])) then cislo_hrany:=1;
  if ((Nod[i,2] = element_trojuhelnik[k,2])and
            (Nod[i,1] = element_trojuhelnik[k,3])) then cislo_hrany:=2;
  if ((Nod[i,2] = element_trojuhelnik[k,3])and
            (Nod[i,1] = element_trojuhelnik[k,1])) then cislo_hrany:=3;
 end;
  seznam_oblasti[index_oblasti][ix]:=
  Format('ELEMENT %8d ARETE %2d',[cislo_elementu,cislo_hrany]);

  for l:=1 to element_ctverec do
  begin
  if((Nod[i,1]=element ctyruhelnik [l,1])or(Nod[i,1]=element_ ctyruhelnik [l,2]) or
  (Nod[i,1]=element_ ctyruhelnik [l,3])or(Nod[i,1]=element_ ctyruhelnik [l,4]))and
  ((Nod[i,2]=element_ ctyruhelnik [l,1])or(Nod[i,2]=element_ ctyruhelnik [l,2])or
  (Nod[i,2]=element_ ctyruhelnik [l,3])or(Nod[i,2]=element_ ctyruhelnik [l,4])))
  then cislo_elementu:=k;

  if ((Nod[i,1] = element_ ctyruhelnik [l,1])and
            (Nod[i,2] = element_ ctyruhelnik [l,2])) then cislo_hrany:=1;
  if ((Nod[i,1] = element_ ctyruhelnik [l,2])and
            (Nod[i,2] = element_ ctyruhelnik [l,3])) then cislo_hrany:=2;
  if ((Nod[i,1] = element_ctyrsten[l,3])and
            (Nod[i,2] = element_ ctyruhelnik [l,4])) then cislo_hrany:=3;
  if ((Nod[i,1] = element ctyruhelnik [k,4])and
            (Nod[i,2] = element_ ctyruhelnik [l,1])) then cislo_hrany:=4;
  if ((Nod[i,2] = element_ ctyruhelnik [l,1])and
            (Nod[i,1] = element_ ctyruhelnik [l,2])) then cislo_hrany:=1;
  if ((Nod[i,2] = element_ ctyruhelnik [l,2])and
            (Nod[i,1] = element_ ctyruhelnik [l,3])) then cislo_hrany:=2;
  if ((Nod[i,2] = element_ ctyruhelnik [l,3])and
            (Nod[i,1] = element_ ctyruhelnik [l,4])) then cislo_hrany:=3;
  if ((Nod[i,2] = element_ ctyruhelnik [k,4])and
            (Nod[i,1] = element_ ctyruhelnik [l,1])) then cislo_hrany:=4;
  end;
     seznam_oblasti[index_oblasti][ix]:=
    Format('ELEMENT %8d ARETE %2d',[cislo_elementu,cislo_hrany]);
 end;

   2:
 begin
  for k:=1 to pocet_elementu do
  begin
  if( (Nod[i,1]=element_trojuhelnik[k,1])and (Nod[i,2]=element_trojuhelnik[k,2])
  and (Nod[i,3] =element_trojuhelnik[k,3]) ) then cislo_elementu:=k;
  end;
  seznam_oblasti[index_oblasti][ix]:=Format('ELEMENT %8d ',[cislo_elementu]);
 end;
```

```
 3:
  begin
  for k:=1 to pocet_elementu do
  begin
  iff (Nod[i,1]=element_ ctyruhelnik [k,1])and (Nod[i,2]=element_ ctyruhelnik [k,2])
  and (Nod[i,3]=element_ ctyruhelnik [k,3]) and (Nod[i,4]=element_ ctyruhelnik [k,4]) )
  then cislo_elementu:=k;
  end;
  seznam_oblasti[index_oblasti][ix]:=Format('ELEMENT %8d ',[cislo_elementu]);
 end;

   end;
end;
end;

Procedure Tkonvertor.vypis_Elementu(var f_maill : textfile);
var
i:longint;
j:byte;
N:longint;
fmt1,fmt2:string;
pomocna1: array [1..MAX_VRCHOLU_NA_ELEMENT] of longint;
begin
 fmt1:='%' + POCET_ZNAKU_REAL + '.' + PLATNYCH_CIFER + 'e';
 fmt1:=fmt1+fmt1;
 for i:=1 to pocet_elementu do

  begin
   case Typ[i] of
          2: begin  // trojuhlenik
            for j:=1 to 3 do
            begin
            N:=Nod[i,j];

            writeln(f_maill,format(fmt1,[Souradnice_x[N],Souradnice_y[N]]));
            pomocna1[j]:=N;
            end;
            fmt2:=FORMAT_MSH_INTEGER;
            fmt2:=fmt2+fmt2+fmt2;
            writeln(f_maill,format(fmt2,[pomocna1[1],pomocna1[2]
                              ,pomocna1[3] ]));
            end;
           3:begin
             writeln('Not yet implemented');
             Halt(1);
            (*writeln(f_maill,'*');
            writeln(f_maill,HLAVICKA_RADEK10, typ_elementu,'',
            Definice_ ctyruhelnik, ':      ', element_ctverec ,'      ',
            HLAVICKA_RADEK10a);
            for j:=1 to 4 do
            begin
            N:=Nod[i,j];
            writeln(f_maill,format(fmt1,[Souradnice_x[N],Souradnice_y[N]])) ;
            pomocna1[j]:=N;
            end;
            fmt2:=FORMAT_MSH_INTEGER;
            fmt2:=fmt2+fmt2+fmt2+fmt2;
            writeln(f_maill,format(fmt2,[pomocna1[1] , pomocna1[2]
                              , pomocna1[3]  ,pomocna1[4] ]));*)
            end;
```

```
    end
  end;

end;

procedure Tkonvertor.hlavicka_oblasti(var f_maill : textfile);
begin
writeln(f_maill , HLAVICKA_KOMENTAR);
writeln(f_maill , HLAVICKA_KOMENTAR_LINKA);
writeln(f_maill , HLAVICKA_RADEK11);
writeln(f_maill , HLAVICKA_KOMENTAR_LINKA);
writeln(f_maill , HLAVICKA_KOMENTAR);

end;

procedure Tkonvertor.nacteni_oblasti(var f_geo : textfile);
var
i:integer;
begin
 i:=0;
 repeat
  readln(f_geo,radek);
   inc(i);
   jmeno:=TrimRightSet(TrimRightSet(radek,[':','0'..'9',' ']),['=',' ']);
   cislo:=StrToInt(
    TrimSet(TrimLeftSet(radek,[' ','a'..'z','A'..'Z','1'..'9']),['=',' ',':']) );
   oblasti[i].jmeno:=jmeno;
   oblasti[i].cislo:=cislo;
  until EOF(f_geo);

  pocet_oblasti:=i;
end;

procedure Tkonvertor.vypis_oblasti(var f_maill :textfile);
var i,j:longint;
begin
 for i:=1 to pocet_oblasti do
 begin
  writeln(f_maill, 'DOMAINE ',"",oblasti[i].jmeno,"",' (Boundary part No. ',
   oblasti[i].cislo,')');
  for j:=1 to N_polozek_oblasti[i] do
  begin
  writeln(f_maill,seznam_oblasti[i,j]);
  end;
  writeln(f_maill, HLAVICKA_KOMENTAR) ;
 end;
 writeln(f_maill, HLAVICKA_RADEK12);
end;

procedure Tkonvertor.konverze(var jmeno_soubor_msh : TFileName);
begin
  System.Assign(file_msh,jmeno_soubor_msh);
  System.Assign(file_geo,jmeno_soubor_domains);
  jmeno_maill:=ChangeFileExt(jmeno_soubor_msh, '.maill');
  System.Assign(file_maill,ADRESAR_VYSTUP+jmeno_maill);
  writeln('Processing ',jmeno_maill,'..');
  Reset(file_msh);
  Reset(file_geo);
  Rewrite(file_maill);
```

```
 try
  nacteni_Nodu(file_msh);
  nacteni_Elementu(file_msh);
  vytvorit_Hlavicku(file_maill);
  vypis_Elementu(file_maill);
  hlavicka_oblasti( file_maill);
  nacteni_oblasti(file_geo);
  rozdeleni_dle_oblasti();
  vypis_oblasti(file_maill);
 finally
  Closefile(file_maill);
  CloseFile(file_msh);
  CloseFile(file_geo);
 end;
 Write('..OK. ');
end;

procedure TKonvertor.DoRun;
var t_start,t_end:TdateTime;
begin
 if HasOption('h','help') or (ParamCount <> 2) then begin
  WriteHelp;
  Halt;
 end
 else begin;
  jmeno_soubor_msh:=ParamStr(1);
  jmeno_soubor_domains:=ParamStr(2)
 end;

 t_start:=Now;
 Konverze(jmeno_soubor_msh);
 t_end:=Now;
 writeln(Format('Elapsed time %.2f seconds.',
  [MilliSecondSpan(t_start,t_end)/1000]));
 Terminate;  // set terminated flag
end;

constructor TKonvertor.Create(TheOwner: TComponent);
begin
 inherited Create(TheOwner);
 StopOnException:=True;
end;

destructor TKonvertor.Destroy;
begin
 inherited Destroy;
end;

procedure TKonvertor.WriteHelp;
begin
 { add your help code here }
 writeln('Usage: ','msh2maill -h .. get help');
 writeln('       ',
  'msh2maill f_msh f_domains .. convert f_msh + f_domains into ./f.maill');
 writeln;
 writeln('Remarks: ');
 writeln('  * files must exist, target file *.maill is overwritten');
 writeln('  * file containing domains must include domain specifications only');
 writeln('    (no comments etc.)');
end;
```

```
var
  Application: TKonvertor;
begin
  Application:=TKonvertor.Create(nil);
  Application.Title:='msh2maill';
  Application.Run;
  Application.Free;
end.
```