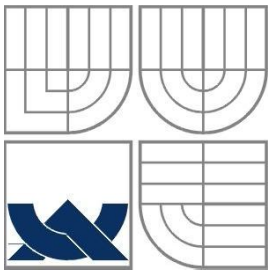
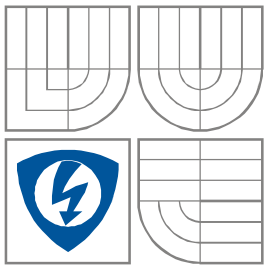


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

PROGRAMOVÉ PROSTŘEDKY PRO SIMULACI A VIZUALIZACI ELEKTROMAGNETICKÝCH STRUKTUR

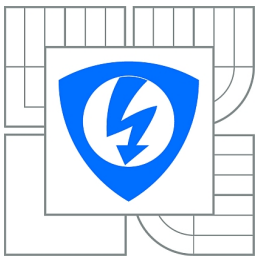
SOFTWARE PLATFORM FOR SIMULATION AND VISUALIZATION OF ELECTROMAGNETIC STRUCTURES

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE Bc. Rostislav Nunvář
AUTHOR

VEDOUCÍ PRÁCE Ing. Michal Pokorný
SUPERVISOR

BRNO, 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Diplomová práce

magisterský navazující studijní obor
Elektronika a sdělovací technika

Student:	Bc. Rostislav Nunvář	***ID:***	98490
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Programové prostředky pro simulaci a vizualizaci elektromagnetických struktur

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s volně dostupnými programovými prostředky pro numerickou analýzu fyzikálních problémů metodou konečných prvků. Prostudujte možnosti dostupných generátorů sítí, řešičů a vizualizačních programů. Vyberte nejvhodnější sadu programů tvořící kompletní simulační prostředí. To znamená, že musí obsahovat graficky editor struktur, generátor sítě, řešič a vizuální zpracování výstupů. V takto vytvořeném simulačním prostředí implementujte sadu ukázkových úloh řešení elektromagnetických struktur zadaných vedoucím práce a vypracujte detailní dokumentaci. Správnost analýz ověřte ve vybraném komerčním programu.

DOPORUČENÁ LITERATURA:

[1] RIBES, A., CAREMOLI, C., Salome platform component model for numerical simulation, Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International, Volume 2, 2007, p. 553-564.

Termín zadání:	7.2.2011	***Termín odevzdání:***	20.5.2011

Vedoucí práce: Ing. Michal Pokorný

prof. Dr. Ing. Zbyněk Raida
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Obsahem diplomové práce je seznámit se s volně šiřitelným software na bázi metody konečných prvků a prostudovat možnosti implementace volně šiřitelných knihoven řešičů, generátorů sítí a vizualizačních nástrojů. Následně jsou zde postupně prezentovány jednotlivé moduly dílčích částí simulace. Dále je vybrán vhodný řetězec OpenSource nástrojů, tvořící kompletní simulační prostředí pro práci s metodou konečných prvků. V další části práce je vypracováno několik dílčích úloh pro řešení elektromagnetických struktur.

Klíčová slova

SALOME PLATFORM, metoda konečných prvků, GETDP, OpenSource, gmsh

Abstract

The content of master´s thesis is to introduce with OpenSource software, which use finite element method and to study facilities implementation of free accessible library solvers, meshes and visualization tools. Further there are individual modules introduced. Is chosen suitable combination of OpenSource tools, which create full simulation facility for finite element method. In the conclusion, several electromagnetic examples are solved.

Keywords

SALOME PLATFORM, finite element method, GETDP, OpenSource, gmsh

Bibliografická citace

NUNVÁŘ, R. *Programové prostředky pro simulaci a vizualizaci elektromagnetických struktur*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 64 s. Vedoucí diplomové práce Ing. Michal Pokorný.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma Programové prostředky pro simulaci a vizualizaci elektromagnetických struktur jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne..... .....

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce ing. Michalovi Pokornému za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne ....

podpis autor

Obsah

Seznam obrázků

1 Úvod

Každý den obklopuje člověka množství předmětů, sloužících k uspokojování lidských potřeb a zjednodušující každodenní práci. Každý předmět musí však absolvovat dlouhou cestu, nežli se stane pro člověka užitečným. Při výrobě je na předmět kladeno mnoho požadavků, jako jsou jeho jednoduchost, dlouhá životnost, minimální vliv na životní prostředí, zvolení optimálního materiálu, testování atd. Ne vždy nám však podmínky umožňují jeho reálné zkoumání. Mohou nastat případy, kdy dané prostředí nejsme schopní vytvořit (chování těles při cestování vesmírem), nebo pokusy jsou velice nebezpečné (neřízené štěpení jader uranu, odolnost přehrady před tlakem vody), či případné zkušební testy jsou finančně velmi náročné (nárazové testy automobilů). S postupem času se tyto nároky stále zvyšují, což vede k nutnosti návrhu a testování předmětů ve virtuálním prostředí. Toto virtuální prostředí nám mohou poskytnout různé počítačové simulace, navržené pro danou oblast působení navrhovaného předmětu.

V průběhu několika posledních let došlo k ohromnému pokroku v oblasti výpočetní techniky. Výkon dnešních výpočetních zařízení byl ještě před několika lety téměř nepředstavitelný. S tímto rozmachem výpočetní techniky se před námi otevírají nové možnosti použití a to v oblasti neustále se vyvíjejícího simulačního softwaru.

Hlavním úkolem simulačního softwaru je na základě známých zákonitostí z fyziky, matematiky, chemie a v neposlední řadě také ze znalosti předešlých zkušeností, vytvořit zobecněný model, který bude řešit danou problematiku návrhu předmětu, nebo zkoumat jeho vlastnosti. Vložíme-li tedy do programu vstupní data, která máme pevně daná (rozměry, struktura materiálu, okrajové podmínky), vytvoříme matematický model fyzikálního problému. Program za použití numerických metod vypočte řešení této matematické reprezentace. V rámci postprocesingu můžeme vyčíslit a vizualizovat hledané veličiny. Tento výsledek je nakonec nutné porovnat se skutečností a provést zpětnou analýzu, zdali jsme se při simulaci nedopustili chyby. Pokud se výsledek se skutečností neshoduje, je nutné chybu analyzovat, zobecnit a tento poznatek se snažit při dalším návrhu využít pro náš prospěch.

V dnešní době se simulační software využívá v různých odvětvích, jako jsou stavitelství, letectví, strojírenství, elektromagnetismus, proudění a další. Je nepostradatelnou součástí výrobního procesu a pomáhá snižovat čas a v důsledku především cenu výsledného předmětu. Na trhu je v dnešní době již ohromné množství simulačního softwaru. Nalezneme zde jak verze komerční, tak i verze navržené pro volně šiřitelnou platformu Linux. Komerční verze jsou obvykle vytvářeny odborníky, a proto je zaručena vysoká přesnost výsledků. Programy při výpočtech využívají různé numerické metody. V této práci se omezím na metodu konečných prvků a také se zmíním a metodě konečných diferencí.

2 Matematické modely

Již řadu let představují numerické metody mocný výpočetní nástroj. Využívají se především pro simulační programy a řešení námi požadovaného problému. Díky rozvoji výpočetní techniky bylo možné vytvořit výpočetní metody, které jsou aplikované na velmi složité geometrické tvary. Těmto geometrickým tvarům je také nutné nadefinovat vhodné okrajové podmínky, aby se daný problém co nejvíce přiblížil reálnému problému. V dnešní době převážná většina simulačních programů používá matematický model, využívající metodu konečných prvků nebo metodu konečných diferencí.

2.1 *Metoda konečných prvků (MKP)*

Vývoj metody konečných prvků započal již na počátku padesátých let 20. století. Velký význam měla tato metoda především v oboru stavebního inženýrství. Metoda konečných prvků je v současnosti považována za nejúčinnější nástroj pro řešení problémů popsaných diferenciálními rovnicemi. Touto metodou lze při dnešních výpočetních možnostech, vyřešit jakoukoli úlohu téměř libovolné geometrie a rozsahu.

Metoda konečných prvků se z matematického hlediska využívá k nalezení aproximovaného řešení parciálních diferenciálních rovnic a integrálních rovnic. Cílem je zcela eliminovat parciální diferenciální rovnice, nebo je převést na jednoduchou ekvivalentní diferenciální rovnici, kterou je možné řešit standardními postupy. Tato metoda je vhodná zejména tam, kde je nutné řešit nepravidelnou strukturu těles [2].

Metoda konečných prvků využívá rovnici, popisující daný problém. Nutností je také definování okrajových podmínek. Základním principem je rozdělení (diskretizace) zkoumané oblasti na prvky (nejčastěji trojúhelníky). Vrcholy těchto prvků nazýváme uzly. Dále provedeme lineární, kvadratickou nebo kubickou aproximaci nad každým konečným prvkem. Aproximovaná funkce je vyjádřena pomocí neznámých hodnot intenzity v uzlech a pomocí známých bázových funkcí. Výhodou této bázové funkce je skutečnost, že nabývá hodnoty jedna v jednom uzlu a nulové hodnoty ve všech ostatních uzlech. Tuto aproximaci dosadíme do řešené rovnice.

Aproximace je pouze přibližná, proto zcela přesně nesplňuje řešenou rovnici. Pro přesně řešení je tedy nutné přičíst reziduum (chybovou funkci). Čím je toto reziduum menší, tím můžeme očekávat přesnější řešení rovnice. Postupným vynásobením rezidua vhodnými váhovými funkcemi a integrováním toho součinu přes celou oblast zkoumaného vlnovodu docílíme minimalizace rezidua. K tomuto účelu lze využít Galerkinovu metodu, při níž zvolíme za váhové funkce bázové funkce všech uzlů, v nichž neznáme hodnotu intenzity. Tím dostaneme N rovnic pro N neznámých. Výstupem této metody je tedy maticová rovnice.

Vyřešením této maticové rovnice získáme hodnoty doposud neznámých uzlových intenzit. Výsledkem řešení této maticové rovnice je vektor vlastních čísel (vektor kvadrátů kritických vlnových čísel jednotlivých vidů) a matice odpovídajících vlastních vektorů (uzlové hodnoty podélných složek intenzit jednotlivých vidů). Následným dosazením uzlových hodnot do aproximace získáme aproximační funkci hledané podélné složky intenzity v každém bodě zkoumané oblasti. Z podélné složky intenzity můžeme dále vypočítat všechny ostatní složky vektorů intenzity elektrického i magnetického pole [21].

2.2 *Metoda konečných diferencí (MKD)*

Metoda konečných diferencí je numerická metoda, sloužící k aproximaci složitých diferenciálních rovnic. Podstatou metody konečných diferencí je pokrytí oblasti, v níž hledáme řešení diferenciální rovnice, sítí, která se skládá z konečného počtu uzlových bodů. V každém bodě sítě se nahradí derivace v těchto uzlových bodech příslušnými diferencemi, tj. lineárními kombinacemi funkčních hodnot v okolních bodech. V závislosti na tom, zda volíme diference dopředné či zpětné, dostáváme různé typy metody sítí (metody explicitní, implicitní). Po záměně derivací diferencemi ve všech uzlových bodech dostáváme soustavu lineárních algebraických rovnic s neznámými hodnotami posunů v těchto uzlových bodech [9].

2.3 *Srovnání obou numerických metod*

Rozdíly mezi MKP a MKD jsou následující:

- Vynikající schopností MKP je schopnost relativně snadno řešit velice komplikované geometrie. Oproti tomu MKD jsou vázány na pravoúhelníkové oblasti, popř. oblasti z nich odvozené. Výhodnou vlastností MKD je její snadná implementace.
- Za bázové funkce lze zvolit po částech konstantní funkce nebo Diracovy delta funkce. V obou přístupech probíhá aproximace na celé oblasti, ale ta nemusí být spojitá. Popř. lze definovat speciální funkci jen na určité oblasti ovšem s tím důsledkem, že spojitý diferenciální operátor již pozbývá smyslu a zvolený přístup již nelze považovat za MKP.
- Kvalita aproximace MKP je často vyšší ve srovnání s MKD, ale na druhou stranu je silně závislá na typu úlohy a lze sestavit příklady, kde naopak bude lepší aproximace u MKD.
- MKP s porovnáním s MKD poskytuje přesnější nahrazení geometricky složitých tvarů a zadaných okrajových podmínek v řešené oblasti [3].

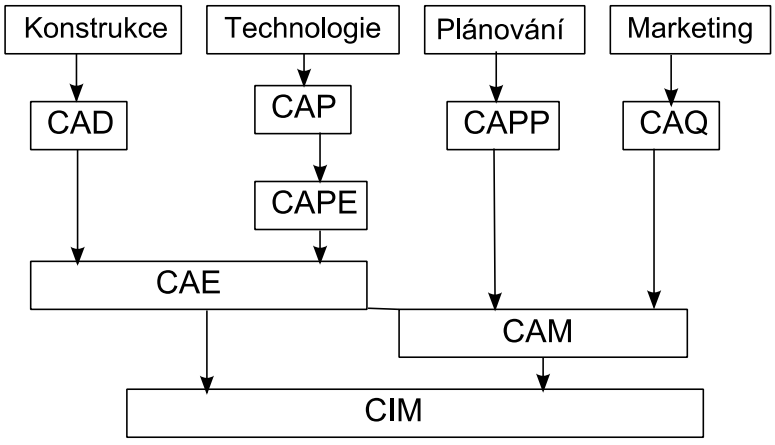
Použití numerických metod je velice všestranné. Mnoho simulací je také zaměřeno například na oblast proudění vzduchu (tekutiny) okolo těles nebo vzduchotechniku. Pevnostní simulace převážně používají pevnostní výpočty, které umožňují simulovat nárazové testy.

3 CAD systémy na bázi Open Source

CAD (Computer Aided Design) software je obecný název pro programové nástroje, které se využívají v počátečních etapách výrobního procesu. Jedná se především o vývoj, konstrukci a technologickou přípravu pro výrobu. CAD je však pouze jednou z mnoha součástí výrobního cyklu. Zkratka je odvozená od označení CAx, kde CA znamená počítačová podpora. CAx technologie tedy spojuje maximální využití výpočetní techniky a programového vybavení, které uživateli (konstruktér, technolog) umožňuje řešení úloh, které souvisí s výrobním procesem. CAx technologii je možné rozdělit do následujících oblastí [11]:

- CIM – Computer Integrated Manufacturing
- CAM – Computer Aided manufacturing
- CAE – Computer Aided Engineering
- CAD – Computer Aided Design
- CAPE – Computer Aided Production Engineering
- CAP – Computer Aided Programming
- CAPP – Computer Aided Process Planning
- CAQ – Computer Aided Quality

Jejich souvislost je možné popsat následujícím Obr. 1.



Obr. 1 *Souvislost CAx systémů.*

CAD systémy se stále vyvíjí a současně zdokonalují s vývojem výpočetní techniky.V této práci se nadále budu zabývat již pouze systémy CAD.

3.1 *OpenSource vs komerční software*

Na trhu je možné nalézt kromě komerčního softwaru také tzv. Open Source software. Komerční software lze chápat jako software, který je možné využívat pouze při jeho zakoupení. Instalace komerčního software na více počítačů je omezena počtem licencí a modifikace zdrojového kódu programu je vyloučena. Zakoupený software je vždy zaměřen na určitou profesní oblast a tvoří kompletní sadu nástrojů pro řešení daného problému. Na vývoji

softwaru pracuje tým odborníků, kteří se snaží vytvořit uživatelsky přátelské a srozumitelné prostředí, které usnadňuje koncovému uživateli práci. Mezi výhody dále patří technická podpora, která nabízí podrobnou dokumentaci programu a možnost zúčastnit se různých školení. Naopak nevýhodou je zpravidla vysoká cena (řádově stovky tisíc korun), proto je tento software dostupný spíše pro velké firmy, které si mohou dovolit zaplatit tuto licenci. Mezi zákazníky patří často velké společnosti automobilového, leteckého, vojenského a stavebního průmyslu. Tento druh software je také často používán vzdělávacími institucemi, jako jsou střední a vysoké školy. Pro vzdělávací účely jsou nabízeny speciální licence se sníženou cenou (tzv. academic licence).

Pokud uživatel není vlastníkem velké společnosti, může se uchýlit k OpenSource softwaru. Tímto označením rozumíme software, který je většinou vytvářen menšími pracovními skupinami či jednotlivci z výzkumného nebo akademického prostředí. Výhodou tohoto programu je, že každý uživatel si může zdrojový kód programu otevřít, různě upravit a přizpůsobit svým potřebám, popřípadě ho může za jistých podmínek dál volně šířit. Ovšem ne každý uživatel je schopný programátor. Nevýhodou je ta skutečnost, že velká část programů neobsahuje kompletní sadu nástrojů. K provedení kompletní analýzy je tedy potřebné využít i dalších dostupných, volně šiřitelných programů. OpenSource programy jsou distribuovány pod nejrůznějšími licencemi, které určují, do jaké míry je daný program volně šiřitelný. V následující Tab. 1 je možné vidět porovnání jednotlivých licencí [14].

Tab. 1 Porovnání druhů licencí [14].

Licence	**Právo upravovat program**	**Právo šířit kopie programu (původní i upravené)**	**Závislý kód musí být vydán pod stejnou licencí**	**Uživatel spustitelné verze má právo na zdrojový kód**	**Uživatel síťového softwaru má právo na zdrojový kód**	**Povinnost propagovat autora**
GPL	*ANO*	*ANO*	*ANO*	*ANO*	NE	NE
Affero GPL	*ANO*	*ANO*	*ANO*	*ANO*	*ANO*	NE
LGPL	*ANO*	*ANO*	NE	ANO	NE	NE
BSD	*ANO*	*ANO*	NE	NE	NE	NE
Původní BSD	*ANO*	*ANO*	NE	NE	NE	*ANO*

Na trhu je v současné době možné nalézt ohromné množství různých simulačních programů. Jako zástupce z kategorie komerčních programů je možné uvést programy COMSOL a ANSYS. Oba programy obsahují kompletní sadu nástrojů pro řešení různých problémů z oblasti mechaniky, magnetismu, proudění tekutin apod. Naopak jako příklad zástupce OpenSource programů je možné uvést například program SALOME a GMSH.

3.1.1 *COMSOL Multiphysics*

Program COMSOL Multiphysics patří mezi profesionální komerční software. Umožňuje řešit značně komplexní fyzikální problémy, které jsou popsané pomocí parciálních diferenciálních rovnic. K řešení využívá metodu konečných prvků. Program umožňuje modelování multifyzikálních dějů v mnoha vývojových oblastech technických i vědeckých oborů [9]. Jednotlivé programové komponenty pro pre-processing, processing a post-processing jsou uživateli zprostředkovány jednotným a intuitivním uživatelským rozhraním, čímž se program COMSOL stal snadno přístupným výukovým nástrojem, aniž by ztratil cokoliv ze své funkčnosti pro profesionální využití [18]. Zahrnuje prostředí pro vytvoření geometrického modelu, generátor sítí, řešič a vizuální zobrazení výsledku. Také nabízí možnost propojení s programem MATLAB a import geometrie z různých CAD systémů. Program také využívá virtualizační technologii JAVA™, díky které můžeme COMSOL používat na libovolném operačním systému.

3.1.2 *ANSYS*

Produkty komerční firmy ANSYS jsou určeny pro analýzu metodou konečných prvků. ANSYS je multifyzikální program, zahrnující strukturální analýzu (statika, dynamika), rázové děje, teplo a proudění, elektromagnetické pole, elektrostatiku, ale také akustiku, lomovou mechaniku a kompozity. ANSYS umožňuje provádět nejen kontrolní výpočty, ale na základě kontrolních výpočtů následně optimalizaci a to jak topologickou, tak i citlivostní analýzy. Nad výpočty je možné provést hodnocení únavy a životnosti [8]. Pro modelování geometrie lze využít vnitřní grafický editor. Pro analýzy složitějších tvarů lze přednostně využít možnost vstupní konverze některých vektorových grafických formátů ze systémů CAD. Pro používání programu ANSYS je nutná pokročilá znalost teorie konečných prvků. Jedná se zejména o podrobné nastavování druhů prvků, rozsahu a přesnosti jednotlivých matic modelu.

3.1.3 *SALOME Platform*

SALOME patří mezi OpenSource simulační programy. Je možné jej volně stáhnout na oficiálních stránkách [10]. SALOME je součástí CAD/CAE technologie, soustřeďující se především na oblast konstruování a simulací. Při numerických simulacích může uživateli nabídnout jak preprocessing, tak i postprocessing. Vše je založeno na přizpůsobitelné architektuře, která umožňuje implementovat různé moduly do simulačního prostředí. Tímto modulem může být jak grafické rozhraní pro návrh geometrických rozměrů, tak i například řešič nebo generátor sítí [13].

Program SALOME byl vyvinut přednostně pro systém Linux a jeho distribuce (Debian, Ubuntu, Mandriva, Fedora Core, Red Hat Enterprise a Scientific Linux). Poslední zkušební verze programu, vydaná koncem roku 2009 je určená také pro systémy Windows. Pro plynulé užívání programu jsou stanoveny následující hardwarové nároky (Tab. 2)

Tab. 2 Hardwarové nároky programu SALOME [10].

Hardware	Minimální konfigurace	Optimální konfigurace
Procesor	Pentium IV	Dual Core
Operační paměť	512 Mb	2 Gb
Prostor na pevném disku	1,5 Gb	5 Gb
Grafická karta (pamět)	64Mb	128 Mb
CD/DVD mechanika	Volitelná	Volitelná

SALOME obsahuje moduly pro řešení dílčích částí simulačního procesu. Při dobré znalosti programovacího jazyka lze moduly rozšířit o vlastní knihovny a ty dále využívat. Hlavními moduly jsou:

KERNEL – obstarává základní funkce celého programu. Jedná se o funkce pro práci s dokumenty (vytváření nového dokumentu, ukládání dokumentu apod.), základní nastavení programu, Python konsole, nápověda programu a další.

GEOM – nabízí funkce pro práci s CAD modely. Umožňuje geometrický návrh (pomocí základních geometrických prvků), import a export CAD modelů různých datových formátů, operace s již vytvořenou geometrií (otáčení, přibližování, barevné odlišení) apod.

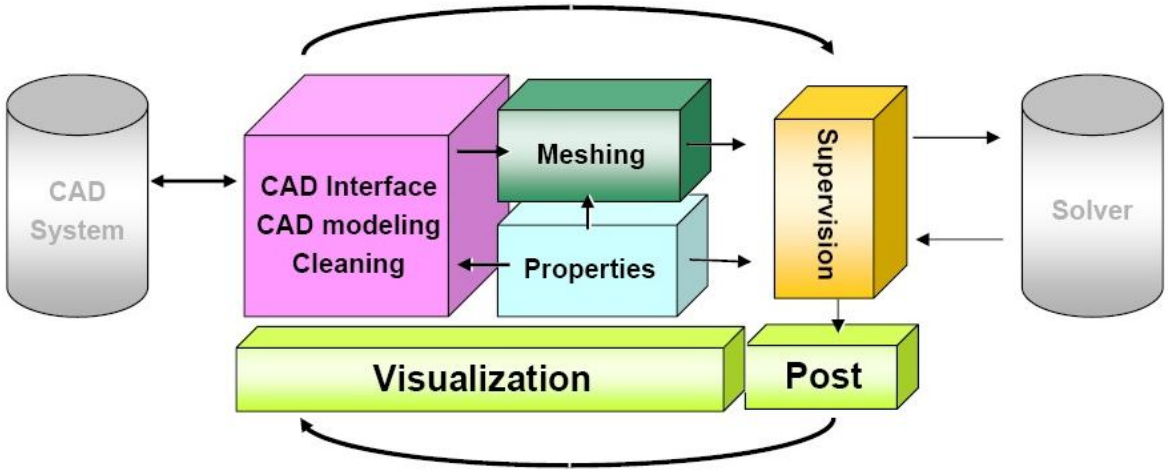
MESH – hlavním úkolem je vytvoření sítě konečných prvků. Lze využít různých algoritmů (Triangulation, Quadrangle, Hexahedron). Dále umožňuje vytváření skupin s odlišnou hustotou sítě, informace o množství vytvořených konečných prvků, možnosti importu a exportu, případně modifikaci importované sítě konečných prvků.

VISU – nástroj sloužící k 2D a 3D grafické representaci výsledků simulace. Nabízí zobrazení jednotlivých povrchů odlišnou barvou, vektorové a skalární zobrazení, animace a vytváření grafů.

4 Postup řešení simulací

Simulační programy postupují při řešení daného problému již podle předem připraveného schématu, který je možné vidět na Obr. 2. Základním krokem je vytvoření geometrické tvaru, rozdělení geometrie tělesa na jednotlivé prvky jednoduchých tvarů (např. trojúhelníky, šestiúhelníky). Tím vznikne diskretizační síť, která je jednoznačně určena množinou prvků a uzlů.

Každá diskretizační síť se může skládat z jednoho nebo i více různých prvků. K vyřešení problému je potom nutné provést výpočet diferenciálních rovnic, které popisují budoucí použití daného tělesa (především je nutné zadat okrajové podmínky, atd.). Tímto krokem se zabývá tzv. řešič. Poté data putují do vizualizačního prostředí, kde se provádí vyhodnocení a případná optimalizace.



Obr. 2 *Schéma popisující propojení částí Salome Platform [7].*

Řešení každé úlohy lze rozdělit do třech základních částí. Nejdříve je nutné navrhnout geometrický model a připravit vstupní data (preprocessing), poté provést vlastní výpočet (processing) a na závěr zpracovat výsledky (postprocessing).

4.1 *Preprocessing*

Cílem preprocesingu je vytvoření geometrického modelu tělesa. Toho je možné dosáhnout například využitím integrovaného modulu GEOM, který je součástí SALOME Platform. Geometrický model je také možné do programu importovat z různých CAD/CAM systému. Při jeho vytváření je nutné brát v úvahu podstatné a nepodstatné tvarové detaily tělesa vzhledem k budoucímu účelu využití. Se složitostí a přibývajícími detaily geometrického tvaru roste také náročnost a doba výpočtů.

Abychom zmenšili velikost geometrického modelu a s tím spojenou náročnost výpočtu, je vhodné využít symetrie tělesa (geometrické symetrie, symetrie materiálu, vlastností). V tomto případě je však nutné nadefinovat na osách symetrie příslušné okrajové podmínky. Pro vytváření diskretizační sítě se využívá tzv. generátorů sítí.

4.2 *Processing*

V této části simulace dochází k vlastnímu výpočtu řešené úlohy pomocí některé z numerických metod. Zásah uživatele je zde z pravidla již minimální. Processing provádí následující úkony:

- Načtení vstupních dat a kontrola jejich správnosti.
- Vytváření prvkových matic.
- Sestavení celkových matic.
- Realizace okrajových podmínek.
- Řešení základní soustavy rovnic.

K výpočtům se využívají především tzv. řešiče .

4.3 *Postprocessing*

Postprocessing slouží k získání vizuální reprezentace výsledků. Moderní simulační systémy již nabízejí propracované grafické prostředí, které umožňuje názorně prezentovat výstupní data. Zejména při simulacích teplotního pole a namáhaných konstrukcích, se využívá barevného rozlišení úrovní míst s různými mechanickými a teplotními vlastnostmi, či různým potenciálem napětí. V důležitých bodech je navíc možné získat přímo konkrétní číselné hodnoty hledané veličiny.

5 Grafický editor struktur

Grafický editor struktur umožňuje vytvořit geometrický tvar zkoumaného tělesa. Jedná se tedy o základní modul pro následné řešení úlohy. Přesnost návrhu geometrického tvaru tělesa má znatelný vliv na konečný výsledek analýzy.

Většina dostupných programů pro modelování již má v sobě začleněn i jednoduchý editor struktur. Avšak úroveň těchto editorů se velmi liší. Je pravidlem, že komerční programy, jako je například ANSYS, používají velice propracovaný grafický editor. Naopak volně dostupné programy se omezují pouze na základní návrh geometrie pomocí jednoduchých tvarů, jako jsou úsečky, přímky, kružnice apod. Velkou výhodou je však to, že je možné importovat geometrie tělesa z jiného programu. Následující kapitoly popisují základní vlastnosti modulů grafických editorů SALOME, GMSH a programu FREECAD.

5.1 *SALOME GEOMETRY*

Program SALOME Platform umožňuje pomocí modulu GEOMETRY vytvářet jednoduchou geometrii těles. Pro kreslení přímek, křivek a oblouků je vhodné použít tzv. sketch. Princip spočívá v zadávání hodnoty souřadnic *x* a *y*, pomocí nichž je možné nakreslit jakýkoli rovinný objekt. Program nabízí také pomocí kreslící lišty na Obr. 3 kreslení prostorových těles, jako jsou čtyřstěny, válce, koule apod.

Obr. 3 *Kreslící lišta.*

Samotné vytváření geometrie je i přes množství funkcí, které program nabízí, dosti složité. Uživatel, zvyklý například na prostředí AUTOCAD, bude velmi zklamán. Není zde umožněno kreslení pomocí pohybu kurzoru myši. Jakýkoli tvar je nutné vytvářet pomocí zadávání souřadnic. Pro vytvoření složitější geometrické struktury je tedy výhodnější použít import z jiného softwaru, kterým je např. INVENTOR nebo AUTOCAD. OpenSource alternativou může být například program FREECAD. Jedná se propracovaný grafický editor, umožňují export nakreslené geometrie do mnoha formátů.

Modul SALOME GEOMETRY nabízí import geometrie pomocí následujících datových formátů:

- BREP
- IGES
- STEP

Je zde však i mnoho užitečných funkcí, které urychlují práci. Mezi tyto funkce patří:

- FACE , vytvoření souvislé plochy mezi zadanými úsečkami.
- EXPLODE , rozdělení zvoleného tělesa na hrany, uzly.
- GROUP , vytváření skupin pro okrajové podmínky a místa s jinou hustotou sítě.
- SKETCH , názorné kreslení pomocí souřadnic *x* a *y*.

5.2 *GMSH GEOMETRY*

Program GMSH nabízí také jednoduchý editor struktur. Kreslení geometrické struktury se z počátku jeví jako uživatelsky nepřívětivé a zdlouhavé. Po pochopení principu je však názorné a smysluplné. Nejdříve je nutné nakreslit body, mezi kterými mohu nakreslit dále úsečky, kruhy a křivky. Pokud pomocí úseček nebo křivek vytvoříme uzavřenou plochu, je možné vytvářet povrchy. Pomocí toho systému může být s každou entitou manipulováno. Je možné ji otáčet, přesunovat, rotovat apod.

Program GMSH podporuje množství vstupních formátů souborů. Mezi nejpoužívanější patří:

- GMSH GEOMETRY
- BREP
- STL
- IGES
- I-DEAS
- STEP

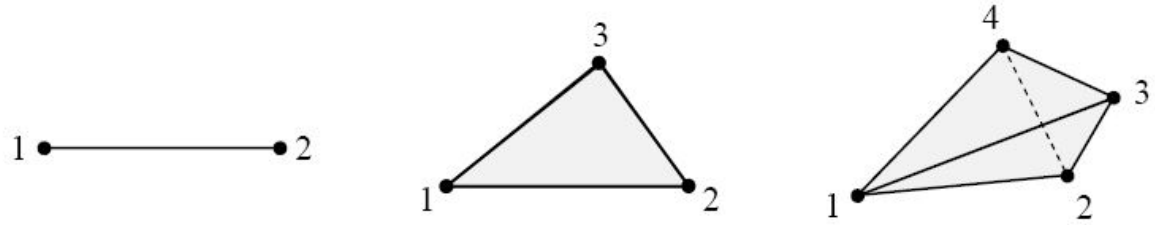
Pro vytvoření složitější geometrické struktury je ale opět výhodnější použít import z jiného softwaru, kterým může být např. FREECAD.

5.3 *FREECAD*

FREECAD je CAD/CAE řešení s otevřenými zdrojovými kódy, které je založené na OpenCasCade, QT a Pythonu. Program obsahuje klíčové vlastnosti jako je srozumitelné a přehledné uživatelské prostředí, nahrávání maker, schopnost pracovat jako server a může být dynamicky zavolán jako rozšíření aplikace. Lze ho využít v mnoha oblastech techniky, jako jsou strojírenství, stavebnictví, ale také ve vědecké a vzdělávací oblasti. Ovládání programu je velmi podobné komerční verzi programu AUTOCAD. Program nabízí velké množství výstupních formátu dat a stává se žádaným grafickým editorem v oblasti OpenSorce software.

6 Generátory sítí

Hlavním úkolem generátorů sítí je rozdělení analyzované oblasti na malé části, tzv. konečné prvky. Každý tento malý konečný prvek již lze snadno matematicky popsat. S ohledem na přesnost analýzy mohou být prvky různě velké, vzájemně na sebe navazují, ale nesmějí se překrývat. V té části, kde potřebujeme detailněji prozkoumat danou oblast, nebo v dané oblasti dochází k výrazným změnám, je možné vytvořit jemnější síť. Toho docílíme zmíněnou změnou velikosti prvků.



a) **b)** **c)**

Obr. 4 *Konečné prvky a) úsečka, b) trojúhelník, c) čtyřstěn [17]*

.

Mezi základní tvary prvků lze považovat úsečku (1D), trojúhelník (2D) nebo čtyřstěn (3D) (Obr. 4). Spojováním těchto základních tvarů lze vytvořit další konečné prvky.

6.1 *GMSH*

GMSH patří mezi třídimenzionální generátory sítí. Využívá Delaunayho triangulaci, při níž je žádoucí, aby vytvořené trojúhelníky byly co nejvíce rovnostranné. Také zahrnuje algoritmus pro adaptivní zjemňování sítě (MeshAdapt), který umožňuje změnu jemnosti sítě v průběhu výpočtu. Nejdříve byl program využíván pouze jako generátor sítí. V současné době jsou součástí tohoto programu také moduly pro vytváření geometrie, generátor sítě, řešič a také rutiny pro postprocessing. Proto představuje plnohodnotný CAD nástroj.

GMSH obsahuje vlastní grafické prostředí, které uživateli poskytuje snadnou orientaci a rychlou práci. Samotný generátor sítí je možné vložit jako modul i do grafického prostředí SALOME PLATFORM. Pro pokročilé uživatele také GMSH nabízí ovládání pomocí příkazového řádku. Zdrojový kód je předkompilovaný pro různé verze operačních systémů (pro Unix, Windows a Mac OS) [4] [5].

Výhody programu GMSH:

- Rychlý návrh geometrického tvaru pomocí vložených funkcí, podmínek, atd.
- Generování 1D, 2D a 3D sítí konečných prvků (přímky, trojúhelníky, čtyřúhelníky, čtyřstěny, hranoly, šestistěny a jehlany).
- Specifikace přesné velikosti prvků. GMSH poskytuje různé mechanismy pro kontrolu velikosti prvku v konečné síti prvků. Využívá interpolace velikosti specifikované při návrhu geometrického tvaru, nebo využívá přizpůsobivé síťové pole.

- Spolupracuje s externími řešiči (GMSH nabízí rozhraní pro C++ a PYTHON, které umožňuje snadné rozšíření o další moduly).
- Kvalitní vizuální výstup (umožňuje zobrazit výstupní data ve skalárním, vektorovém nebo tenzorovém formátu). Nabízí export získaných hodnot do různých formátů.
- Umožňuje vytváření animací.
- Pracuje na různých typech operačních systémů.
- Program lze bez problému spustit i na méně výkonných počítačích.

Nevýhody programu GMSH:

- Není vhodný k návrhu geometrických tvarů těles velkých rozměrů. V tomto případě je vhodné vložit binární kód GMSH do externího CAD systému.
- Uživatelské grafické rozhraní zpřístupňuje pouze omezené množství dostupných vlastností.
- Skriptovací jazyk je dosti omezen, nabízí pouze primitivní programovou smyčku ovládání a uživatelských funkcí, bez žádných lokálních proměnných.
- V případě potřeby odstranění chyby je nutné upravit zdrojový kód

6.2 *NETGEN*

Projekt NETGEN začal již v roce 1994 ve formě diplomové práce na univerzitě Linz v Austrálii. Dále v něm bylo pokračováno díky financování tamní univerzity.

NETGEN je automatický generátor sítí pro 2D a 3D algoritmy. Program je distribuován jako OpenSource pod licencí LGPL. Jedná se o samostatný program s grafickým uživatelským rozhraním, který může být také použit jako C++ knihovna pro aplikace třetích stran. Tento generátor sítí je dostupný pro platformy UNIX/LINUX a WINDOWS 98/NT a je dostupný na stránkách výrobce [3]. Je schopný generovat trojúhelníkové a čtyřúhelníkové sítě. Dále obsahuje modul pro síťovou optimalizaci a hierarchické zjemnění sítě.

6.3 *TETGEN*

TetGen je generátor 3D sítí, vhodný především při využití metody konečných prvků nebo metody konečných objemů. Program je volně dostupný na stránkách [19]. Pro jeho spuštění jsou nutné pouze standardní knihovny C++. Pracuje na 32-bit i 64-bit verzích systému. Podporované systémy jsou Unix/Linux, MacOS a Windows. TetGen využívá především algoritmus typu Delaunay.

Kód je psaný v jazyce C++. Z tohoto důvodu může být kompilován do spustitelného souboru, nebo do knihovny, kterou je možné integrovat do různých aplikací.

TETGEN neobsahuje grafické uživatelské rozhraní (GUI). Pro zobrazení výsledků je nutné použít například program TETVIEW, MEDIT nebo GID [6].

6.4 *SALOME MESH*

Modul MESH, který je součástí programu SALOME, umožňuje vytvářet a modifikovat 1D, 2D a 3D diskretizační síť konečných prvků. K tomuto účelu využívá řadu vlastních algoritmů (hexahedron, quadrangle, triangulation, wire discreatization, atd.), nebo lze využít externí algoritmus (BLSURF, tetrahedron NetGen, HDS3D, hexotic). Umožňuje také nastavení parametrů algoritmu (maximální velikost hran prvků, velikost úhlu apod.).

Pro oblasti s různou hustotou konečných prvků lze vytvářet skupiny, na které jsou dále aplikovány odlišné vlastnosti (zbarvení oblasti, změna prvků v dané oblasti, apod). Přínosem jsou také souhrnné informace o celkovém množství konečných prvků, délce hran apod.

Pomocí nástrojů import lze do programu nahrát již vytvořený soubor. Také lze z programu vyexportovat vygenerovanou síť konečných prvků a využít ji v jiném programu. Je nutné použít některý z podporovaných formátů (Tab. 3)

Tab. 3 Podporované formáty pro import/export

Import	MED, UNV, DAT
Export	MED, UNV, DAT, STL

7 Řešiče

Parciální diferenciální rovnice určují vztah mezi funkcí několika proměnných a jejich derivací. Pomocí jedné nebo více parciálních diferenciálních rovnic je možné vyřešit velké množství problémů, se kterými se běžně setkáváme ve fyzice, strojírenství, matematice a jiných technických oborech. V oblasti OpenSource nástrojů lze použít pro vyřešení těchto rovnic některý z následujících programů.

7.1 *FREEFEM++*

FREEFEM patří do skupiny programů, které umožňují řešit parciální diferenciální rovnice. Jedná se o volně šiřitelný program, který je založen na metodě konečných prvků. Řeší fyzikální úlohy jako např. interakce tekutin a pevné látky, které vyžadují interpolaci dat mezi různými sítěmi. Užívá rychlou interpolaci „quadtree“ [2]. Je kompatibilní se všemi OS na bázi UNIX, Windows a MacOS.

7.2 *GETFEM++*

Tento projekt je zaměřen na vývoj výkonné knihovny C++, umožňující využít početní metodu konečných prvků. Cílem je poskytnout knihovnu, která bude umožňovat výpočet jakékoli základní matice (i smíšené metody konečných prvků) na nejrozsáhlejší třídě metod a elementů, libovolné dimenze (ne pouze 2D a 3D dimenze). Umožňuje snadné propojení s programem MATLAB a skriptovacím jazykem PYTHON.

Tato knihovna také nabízí běžné nástroje pro konečné prvky, mezi které patří kompletní postup pro klasické parciální diferenciální rovnice, interpolační metody, okrajové podmínky a základní postprocessing nástroje. Lze také s výhodou využít pro obecné kódy metod konečných prvků, protože nabízí snadnou změnu parametrů.

Program však nenabízí nástroje pro generování sítě, z tohoto důvodu je nezbytné do programu importovat již vytvořenou síť konečných prvků. Podporované formáty sítí jsou GID, GMSH a EMC2 [1].

7.3 *FENICS*

Projekt FENICS se zabývá automatickým řešením diferenciálních rovnic. Program umožňuje pracovat s vygenerovanými sítěmi, ODE a PDE formáty konečných prvků a lineární algebrou. Jedná se o sbírku knihoven, uživatelských rozhraní a řešičů, mezi které patří:

- DOLFIN , C++/Python knihovna pro řešení diferenciálních rovnic.
- FCC , kompilátor pro konečné prvky různých formátů.
- FERARI , optimalizace pro vyhodnocení různých formátů,
- PUFFIN , jednoduchý řešič konečných prvků pro OCTAVE/MATLAB.

DOLFIN je napsán v jazyce C++ a používá jednoduché a intuitivní prostředí. Využívá předpokladů a Krylovy metody [12].

7.4 *ELMER*

Program ELMER je freeware program, vytvořený společností CSC pro modelování fyzikálních úloh. Vývoj programu začal již v roce 1995 ve spolupráci s finskými univerzitami, výzkumnými ústavy a průmyslem. ELMER obsahuje například modely dynamiky tekutin,

konstrukční mechaniky, elektromagnetismu, tepelné přeměny a zvukové přeměny. Tyto úlohy jsou popsány parciálními diferenciálními rovnicemi, které jsou v programu řešeny pomocí metody konečných prvků.

Řešení modelů parciálních diferenciálních rovnic pomocí řešiče ELMER vyžaduje důkladný popis řešeného problému. Tento popis lze do programu importovat pomocí `sif` souboru, který obsahuje uživatelem připravená vstupní data, určuje umístění potřebných souborů, parametry materiálu, okrajové podmínky, počáteční podmínky a další [6]. Celý program je napsán v jazyce Fortran90, C a C++.

ELMER se skládá z několika různých modulů, které je možné spustit nezávisle na ostatních. Po instalaci programu nám jsou nabídnuty tyto části:

ELMER GUI

- Jedná se o grafické rozhraní, které slouží pro definici geometrie tělesa a následné generování dat pro modul ELMERSOLVER (tzn. vytvoření geometrie, definice okrajových podmínek, fyzikální vlastnosti).
- Schopnost řešit 2D a 3D úlohy.
- Součástí je i generátor sítě, který lze použít pro generování sítě vytvořeného tělesa nebo pro generování sítě tělesa, které bylo vytvořeno jiným programem a naimportováno do ELMER GUI.

ELMER SOLVER

- Zpracovává vstupní data a vygenerovanou síť.
- Umožnuje velké množství úloh (elektrostatika, magnetismus, pružnost, atd.).

ELMER POST

- Jedná se o grafické prostředí pro vizualizaci výstupů z programu ELMER.
- Nabízí barevné vykreslení skalárních polí, vektorová pole a atd.

7.5 *GETDP*

Program GETDP vyvinul Patrick Dular a Christophe Geuzaine na univerzitě v Liège. Jedná se o obecný řešič pro numerické řešení integro-diferenciálních rovnic. Umožňuje výpočty fyzikálních úloh (elektromagnetické, termální, apod.) využitím numerických metod (metoda konečných prvků, integrální metody apod.). Jeho pomocí lze řešit 1D, 2D i 3D dimenze.

Hlavní výhodou GETDP je propojení mezi vnitřní strukturou, definicí vstupního problému a symbolickým vyjádřením řešeného problému. Program je tedy vývojovým nástrojem s ohromnou volností pro vytváření nových funkcí. Nabízí různá uplatnění v oblastech výzkumu, vzdělávání, individuálních projektů a průmyslových studií.

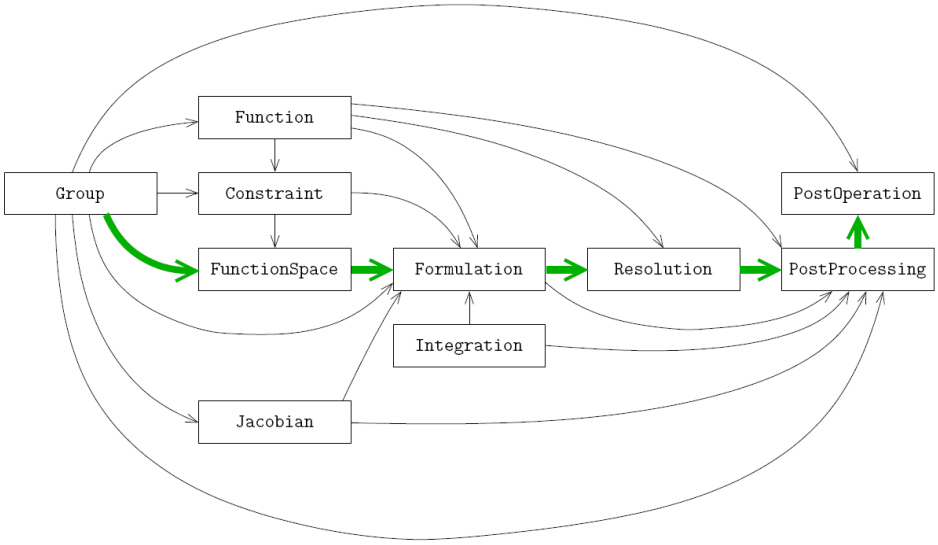
Program neposkytuje žádné grafické prostředí pro vytváření vstupních souborů a pro definici řešení problému.

GETDP je volně ke stažení [15] ze stránek autora programu. Aktuální stabilní verze je dostupná pro operační systémy Windows (XP/Vista/Win 7), Linux a MacOS X.

.

7.5.1 *Struktura programu*

Při definici problému se postupuje dle Obr. 5. Každý blok zobrazené struktury charakterizuje část řešeného problému, pomocné funkce, formulace problémů atd. Pro psaní kódu lze využít jakýkoli textový editor.



Obr. 5 *Propojení struktury GETDP [15].*

Pro spuštění simulace se nám nabízejí dvě možnosti. První možností je využít příkazový řádek, kde lze pomocí sledu příkazů

```
getdp filename options
```

provést spuštění simulace, kde `filename` představuje ASCII soubor s definicí problému. Tento soubor může v sobě zahrnovat i odkazy do ostatních souborů. Vstupní soubor tedy obsahuje definici problému a musí být uložen s příponou `*.pro`. V průběhu výpočtu program vygeneruje další potřebné soubory, které je možné později využít například na vizuální zobrazení výsledků.

Druhou možností spuštění simulace je využití grafického prostředí programu GMSH. Po spuštění programu se přepneme do modulu *Solver* a poté klikneme na nabídku *GETDP*. Spustí se nám dialogové okno, ve kterém do druhého řádku zadáme námi vytvořený vstupní soubor s příponou `*.pro` a do třetího řádku vložíme název souboru s vytvořenou diskretizační sítí s příponou `*.msh`. Spodní část okna nabízí tlačítka pro spuštění dílčích kroků simulace (*Pre, Cal* s *Pos)*. Výsledkem budou opět výstupní soubory, které je možné dále využít.

8 Vizualizace

Vizualizace nám umožňuje názorné grafické znázornění výsledků simulací. Také poskytuje člověku souhrnný přehled a určité povědomí o správnosti simulace.

Při hledání vhodného vizualizačního nástroje záleží na povaze zobrazovaných dat. Pokud uživatele zajímá vlastní struktura sítě (například si chce ověřit, zdali generování sítě dopadlo dle očekávání), měl by nástroj umožňovat pohled na síť z různých pozic a úhlů, měl by nástroj obsahovat například propracované možnosti přiblížení požadované oblasti a být schopen zobrazit i detaily sítě. Pokud je cílem zobrazit vypočtené výsledky a namapovat je na povrch sítě, výhodné je zobrazení v přehledné formě, jakou může být dostatečně rozlišitelná barevná škála [16].

Na trhu existuje velké množství simulačních programů, které přímo v sobě zahrnují jednoduchý vizualizační nástroj. Je však i spousta takových, které vytvoří pouze výstup v podobě datového souboru. V tomto případě je nutné pro kontrolu výsledku simulace využít některý z vizualizačních nástrojů.

8.1 *PARAVIEW*

Program PARAVIEW patří mezi velmi pokročilé vizualizační OpenSource nástroje. Na vývoji tohoto programu se podílí řada společností, jako jsou KitWare, Computational Simulation Software a podobné. Program je distribuován pro tři hlavní platformy:

- Windows
- Linux
- Mac OS X

PARAVIEW je také navržen pro běh na paralelních počítačích se sdílenou nebo distribuovanou pamětí. Tímto lze dosáhnout kratšího času při vizualizaci velkých objemů dat. Samozřejmostí je ale také běh programu na klasickém stolním počítači.

Velkou výhodou programu je ohromné množství podporovaných formátů [11] a také spolupráce s datovými formáty knihovny VTK. Pro naše účely potřebujeme zobrazit data získaná metodou konečných prvků. K tomuto účelu se využívá barevné rozlišení jednotlivých prvků v závislosti na hodnotě intenzity neznámé veličiny v daném uzlu prvku.

8.2 *GMSH POSTPROCESSING*

Součástí programu GMSH je také postprocessing modul. Tento modul umožňuje pracovat se skalárními, vektorovými nebo tenzorovými daty, které jsou spjaty s geometrií a sítí konečných prvků. Vstupní data mohou být importována v různých formátech. Mezi nejpoužívanější formáty patří `*.pos`, MSH (ASCII nebo binární soubor s příponou *.msh) a několik dalších formátů, které byly vytvořeny programy třetích stran.

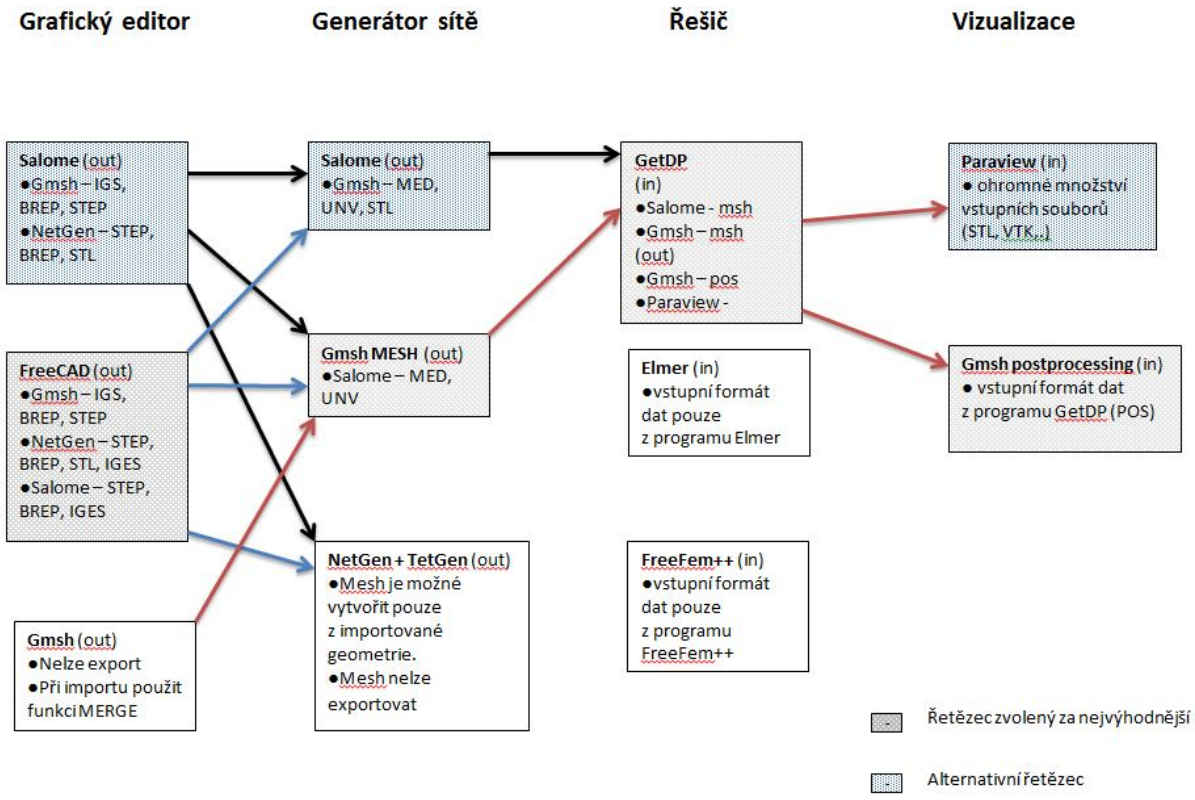
Skalární pole je možné zobrazit jako souhrn přímek, ploch nebo barevných map, zatímco vektorové pole může být reprezentováno také pomocí 3D nebo posunutých map. Pro programátory je tu také možnost veškeré příkazy a nastavení použít pomocí skriptovacího jazyku. Tímto je umožněna naprostá automatizace vizualizačního procesu.

9 Řetězec programů a možnosti propojení

V průběhu zkoumání různých OpenSource programů jsem zjistil, že velmi důležitou vlastností programu je i schopnost exportovat dílčí část procesu (geometrie objektu, vygenerovaná síť konečných prvků, výstup řešiče) do vhodného formátu. V dnešní době se používá ohromné množství formátů dat, ne každý program s nimi však umí pracovat.

Zaměřil jsem se na několik vybraných programů a prozkoumal možnosti vzájemné výměny dat. Na Obr. 6 je zobrazen řetězec programů, tvořících kompletní simulační prostředí. Z grafických editorů se jeví jako nejvhodnější použít FREECAD, pro vygenerování sítě konečných prvků nám poslouží MESH modul programu GMSH, nebo SALOME. Řešič je velmi specifická část řetězce a tvoří hlavní část simulačního procesu. Jako řešič, který se ze zkoumané množiny programů jevil jako nejvhodnější, byl vybrán GETDP. Specifikace řešeného problému je definována uživatelem v oddělených zdrojových souborech. Tyto soubory jsou v průběhu simulace postupně načítány a realizují se definované operace. Ohromnou výhodou tohoto programu je aktivní podpora ze strany autora programu a několik elementárních příkladů, které slouží k pochopení problematiky. Poslední částí simulace je vizualizační prostředí, v kterém je graficky znázorněn výsledek simulace. Zde je možné požít modul programu GMSH (Gmsh postprocessing), nebo velmi dobře propracovaný nástroj PARAVIEW.

U každého uvedeného programu nebo modulu (Obr. 6) jsou vypsány formáty výstupních souborů, které je schopen program exportovat. Tyto formáty dat lze bez problému použít pro import souborů do jiného programu.



Obr. 6 *Kompletní řetězec při simulaci MKP a formáty dat.*

10 Řešení elementárních příkladů

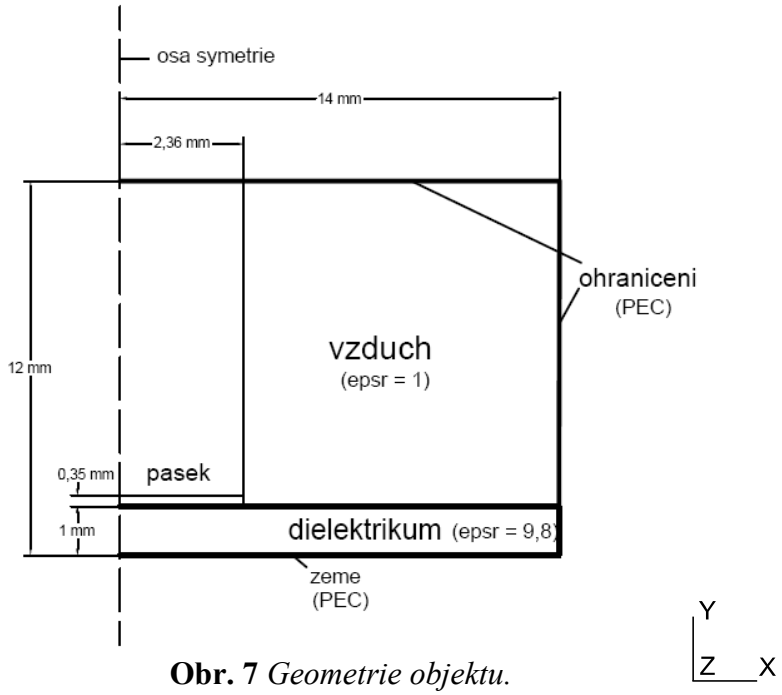
Obsahem následujících kapitol je ukázkové řešení několika elementárních příkladů z oblasti elektromagnetismu. Byly zpracovány vybrané úlohy z počítačových cvičení k předmětu Elektromagnetické vlny, vedení a antény [18]. Jedná se o úlohy:

- rozložení elektrického potenciálu v okolí mikropásku
- šíření elektromagnetické vlny (časová a frekvenční oblast)
- rozložení pole ve vlnovodu, kritické kmitočty vidů
- rozložení pole po průchodu štěrbinou a dielektrickou překážkou

Pro návrh geometrie je vhodné využít grafických editorů programů SALOME nebo GMSH. Další dílčí kroky budou již provedeny v modulech programu GMSH a pomocí řešiče GETDP. U každého příkladu bude uveden stručný rozbor řešeného problému a následný popis nejdůležitějších částí zdrojového kódu řešiče GETDP. Kompletní zdrojový kód pro řešení daných úloh bude k dispozici na přiloženém médiu. Na závěr každého příkladu bude uvedeno srovnání výsledků, dosažených použitím komerčního software COMSOL Multiphysics.

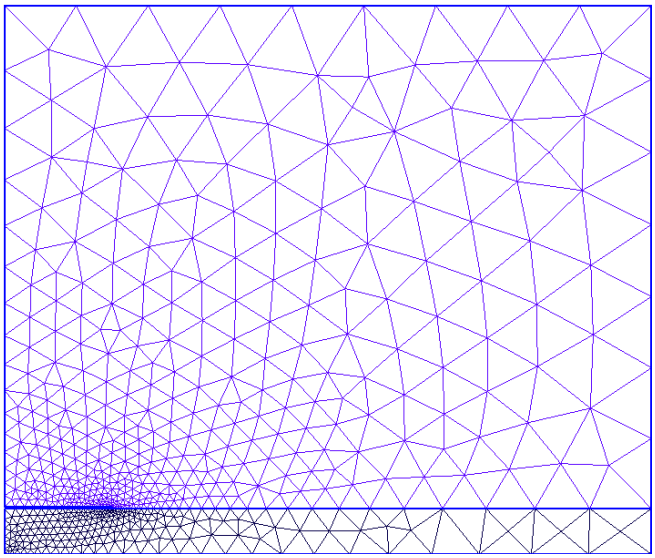
10.1 *Rozložení elektrostatického pole v okolí mikropásku*

Řešeným příkladem (Obr. 7) je rozložení elektrostatického pole v okolí mikropásku (*pasek*). Mikropásek je obklopen ze spodní strany dielektrikem s hodnotou relativní permitivity $\varepsilon_r = 9{,}8$. Prostor nad mikropáskem je vyplněn vzduchem o relativní permitivitě $\varepsilon_r = 1$. Aby bylo dále možné problém numericky modelovat, je třeba stanovit okrajové podmínky. Elektrický potenciál v okolí náboje dosahuje nulové hodnoty až téměř v nekonečnu. Z tohoto důvodu je nutné analyzovaný prostor omezit. V našem konkrétním příkladě je toto omezení realizováno nastavením okrajové podmínky elektrického potenciálu na stěnách *ohraniceni* na hodnotu 0 V. Z důvodů ušetření výpočetního času a zjednodušení simulace je výhodné využít osové souměrnosti. Výsledkem je simulace pouze poloviny objektu, což je k pochopení rozložení elektrického potenciálu dostačující. Mikropásek zde představuje zdroj elektrického náboje s hodnotou $Q = 10^{-3}$ Cm^{-1}.



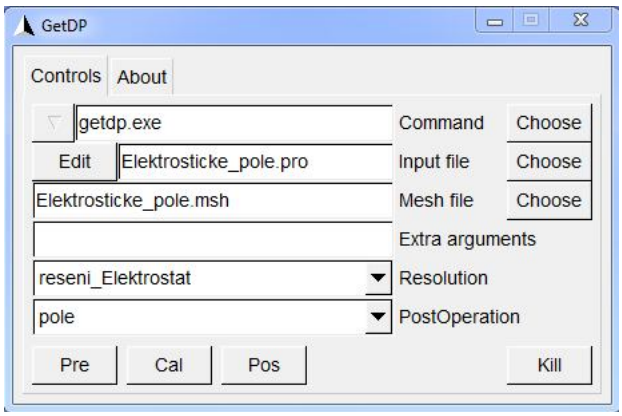
Obr. 7 *Geometrie objektu.*

Nejdříve je nutné vytvořit v programu SALOME PLATFORM nebo GMSH geometrický tvar objektu (Obr. 7). Pokud geometrii objektu vytváříme pomocí grafického editoru v SALOME PLATFORM, je nutné použít export do formátu IGS nebo BREP. Po načtení geometrie v programu GMSH je důležité zvolit položku *File – Merge*. Tento úkon převede geometrii do normované struktury, která je dále snadno zpracovatelná programem GMSH (formát `*.geo`). V dalším kroku pomocí modulu *Gmsh Mesh* provedeme vygenerování sítě konečných prvků (Obr. 8).



Obr. 8 *Síť konečných prvků*

Nyní je nutné stanovit okrajové podmínky a sestavit kód, který popisuje tento problém. Pro vyřešení je použit řešič GETDP. Jeho ovládání je umožněno pomocí příkazového řádku, nebo pomocí grafického modulu programu GMSH (Obr. 9).



Obr. 9 *Grafické ovládání řešiče GETDP*

Vstupními daty je soubor s příponou `*.pro`, který definuje řešený problém. Obsahuje informace o hodnotách permitivit použitých materiálů, okrajových podmínkách, oblastech jejich použití a umístění dalších potřebných souborů s předem připraveným obsahem. Dalším vstupním souborem je vygenerovaná síť konečných prvků, uložená ve formátu `*.msh`.

Výsledek je možné zobrazit také pomocí programu GMSH. Výsledkem našeho příkladu je rozložení elektrostatického pole v okolí mikropásku (Obr. 10).

Na tomto jednoduchém příkladu bude podrobněji vysvětlen postup pro vytváření zdrojového kódu geometrie a definice problému pro řešič GETDP.

Pro použití řešiče GETDP je nutné vytvořit geometrii, která je zapsaná v určité struktuře. Program je poté schopen s jednotlivými prvky snadno pracovat a také zápis je dostatečně srozumitelný pro modifikaci uživatelem. Při importu geometrie z jiného programu je vhodné v programu GMSH použít funkci *File-merge* a soubor poté uložit s příponou `*.geo`. Struktura takového souboru může vypadat následovně:

```
Point(1) = { 0 , 0 , 0 , p0 };

Point(2) = { xBox, 0, 0, pxBox};
```

Tímto zápisem se definují jednotlivé body. Čísla v závorce udávají souřadnice v prostoru *x, y, z*. Pro 2D problém budou souřadnice *z* vždy nulové. Poslední údaj v závorce charakterizuje velikost elementu pro generátor sítě. Je tedy možné v různých oblastech definovat různou hustotu generované sítě konečných prvků. Body se dále spojují do úseček.

```
Line(1) = {1, 2};

Line(2) = {2, 3};
```

Údaje v závorce určují, mezi kterými dvěma body bude úsečka vytvořena. Dále je možné tyto úsečky spojovat a tím vytvářet objekty. Podmínkou je však to, že vytvořený objekt bude uzavřený.

```
Line Loop(14) = {10, -11, 8, 3, 4, 5};
```

Z každé takto vytvořené uzavřené smyčky lze vytvořit plochu.

```
Plane Surface(15) = {14};
```

Tímto způsobem jsme tedy nadefinovali geometrii objektu. Nyní je možné si dále nadefinovat objekty, které využijeme v řešiči. Těmto objektům se říká regiony a udávají informaci o vzájemném propojení jednotlivých elementů.

```
Physical Surface (101) = {15};
Physical Line (120) = {1};
```

Dalším krokem je vytvořit síť konečných prvků. K tomu bude využit nástroj programu GMSH, tedy MESH generátor. Zde je možný výběr mezi 1D, 2D a 3D dimenzí a také lze nastavit algoritmus pro generování sítě (MeshAdapt, Delaunay, Frontal). V řešených úlohách bude využito nestrukturovaného algoritmu MeshAdapt, který je schopen vytvářet trojúhelníkovou síť v závislosti na aktuálních potřebách jemnosti sítě.

Nyní je nutné využít řešič, specifikovat řešený problém a zobrazit výsledky. Zde se tedy zaměříme na definici dílčích kroků, jako jsou vytvoření oblastí, funkcí, definice rovnic, výpočet a v neposlední řadě také zobrazení výsledku.

Způsob, kterým je problém možné řešit, lze nejlépe pochopit z Obr. 5. Prvním krokem je definice tzv. `Group` objektu. Jedná se o oblasti geometrie, které byly definovány v souboru `*.geo` (`Physical Surface`, `Physical Line`). Zde je těmto oblastem přiřazena proměnná a také je možné tyto oblasti různě spojovat do větších celků. Všechny prvky, patřící do skupiny `Group` musí být uzavřeny ve složených závorkách.

```
Group {
            vzduch = Region[101];
            zeme = Region[120];
            sum = Region[{vzduch, dielektrikum}];
      }
```

V našem případě je nutné definovat oblasti:

`vzduch`	prostor nad mikropáskem.
`dielektrikum`	oblast dielektrika.
`zeme`	úsečka, sloužící pro definici okrajové podmínky mezi dvěma prostředími.
`pasek`	oblast mikropásku.
`ohraniceni`	úsečky, tvořící hranice simulovaného prostředí, na které je uplatněna okrajová podmínka.
`sum`	oblast `sum` spojuje regiony `vzduch` a `dielektrikum`. Tím vytváří celkovou oblast, na které bude proveden výpočet.

Následujícím objektem je `Function`. Slouží nám k definování funkcí, konstant, proměnných, vlastností konkrétních oblastí atd. `DefineFunction` umožňuje definovat globální funkci, kterou je možné využít v ostatních objektech programu. Definice konstant je totožná s ostatními programovacími jazyky. Do hranatých závorek se vkládá název oblasti, ke které se bude tato podmínka vztahovat.

```
Function {
            DefineFunction[ epsr ];
            eps0 = 8.854187817e-12;
            epsr[vzduch] = 1.;
            epsr[dielektrikum] = 9.8;
          }
```

Objekt `Constrain` umožňuje definovat okrajové podmínky. Okrajové podmínky se vždy vztahují k určitému regionu. Pokud je nutné definovat okrajových podmínek více, využívá se příkazu `Case`. `Type Assign` udává, že se jedná o Dirichlet okrajovou podmínku (určitému regionu bude přiřazena konkrétní hodnota veličiny).

```
Constraint {

      { Name ElectricScalarPotential; Type Assign;
      Case {
            { Region Region[{zeme, ohraniceni }]; Value 0.; }
            { Region pasek; Value 1.e-3; }
           }
      }

    }
```

`FunctionSpace` je objekt, který je charakterizován typem pole (skalární či vektorové). Jeho hlavním úkolem je vytvořit bázové funkce a postarat se o uplatnění odpovídajících okrajových podmínek. Typ objektu `Form0` je již autorem nadefinovaná funkce, která umožňuje řešit skalární pole pro výpočet rozložení potenciálu. `Function BF_Node` v objektu `BasicFunction` slouží k tomu, aby všechny vypočítané hodnoty rozložení potenciálů byly přiřazeny k odpovídajícím uzlům v daném regionu (v tomto případě se jedná o region `sum`). Výše nadefinované okrajové podmínky jsou zde zahrnuty pomocí výrazu `NameOfConstraint`, který odkazuje do objektu `Constraint`.

```
FunctionSpace {
      { Name SP_electrostatic; Type Form0;
            BasisFunction {
                  {Name sn; NameOfCoef vn; Function BF_Node;
                   Support sum; Entity NodesOf[ All ]; }
                        }

Constraint {

      { NameOfCoef vn; EntityType NodesOf;
        NameOfConstraint ElectricScalarPotential; }
            }
                        }
               }
```

Nejdůležitějším objektem je objekt `Formulation`. Skládá se ze tří částí. Nejprve je pojmenování objektu (`Poisson`) a deklarace jeho typu. `Type FemEquation` definuje, že se jedná o výpočet pomocí metody konečných prvků (umožňuje řešit úlohy také pomocí momentové a integrální metody). `Quantity` definuje neznámou veličinu *v*, kterou je v našem případě rozložení elektrostatického pole. Dále odkazuje na jméno objektu `FunctionSpace`, s kterým je propojena. Poslední částí je `Equation`, do které se zapisují rovnice nebo soustavy rovnic, sloužící pro řešení daného problému.

V obecném tvaru Poissonovy rovnice proměnná *a* představuje difúzní koeficient, *e* označuje rozložení skalární veličiny a *f* reprezentuje zdroj. Pro řešení rozložení elektrostatického pole proměnná *a* představuje permitivitu prostředí, *v* odpovídá rozložení elektrostatického potenciálu a *f* reprezentuje hustotu volného náboje.

$$\operatorname{div}\left(a \cdot \operatorname{grad}(v)\right) = f \tag{1}$$

Formulace této rovnice v programu GETDP je následující

```
epsr[]*Dof{d v}, {d v}
```

,kde `epsr[]` je funkce, obsahující informaci o hodnotách permitivity v jednotlivých regionech geometrie. Výraz `Dof{}` definuje hledanou neznámou veličinu *v*. Výraz *d* označuje derivaci. Vše je opět vztaženo k určitému regionu (`sum`). Při výpočtu jsou využívány objekty `Jacobian` a `Integration`, které budou popsány níže.

```
Formulation {
            { Name Poisson;Type FemEquation;
            Quantity {
                  {Name v; Type Local; NameOfSpace SP_electrostatic;}
                      }
            Equation {
                  Galerkin {[epsr[]*Dof{d v},{d v}];In sum;
                  Jacobian Vol; Integration I1;}
                      }
      }
             }
```

Takto sestavená rovnice, či soustava rovnic je dále řešena pomocí objektu `Resolution`. Ve složitějších případech se definuje několik objektů `Resolution`, které se poté liší svým jménem a také způsobem řešení problému. Zde je umožněno propojení s objektem `Formulation` (definice rovnic pro vyřešení problému) a dále je možné zde upřesnit, zdali se problém bude řešit ve frekvenční oblasti, časové oblasti, případně změnit typ algoritmu (GMRES, CGNR,..).

```
Resolution {
      { Name reseni_Elektrostat;
            System {
                  { Name A;NameOfFormulation Poisson;}
                    }
            Operation {Generate[A];Solve[A];SaveSolution[A];
                     }
       }
            }
```

Objekt `PostProcessing` využívá vypočtenou neznámou veličinu, definovanou v objektu `Formulation`, a pomocí autorem nadefinovaných výrazů či matematických operací umožňuje její následnou úpravu. Takto můžeme tedy zobrazit rozložení elektrostatického pole nebo vektorů, charakterizujících směr největšího růstu potenciálu.

V definici objektu `PostProcessing` je nutné definovat jméno, název objektu `Formulation` a také jméno objektu `System`, který jsme použili. Opět je zde nutné zapsat, na jaký region se má daný příkaz uplatnit.

```
PostProcessing {
{ Name Elektrostat_pole; NameOfFormulation Poisson; NameOfSystem A;
      Quantity {
            { Name v; Value { Local { [ {v} ]; In Sum;Jacobian Vol; }
            }
               }
            }
                }
```

Objekt `PostOperation` slouží k vytvoření souborů s požadovanými daty, které lze využít k zobrazení výsledků. Výsledky můžeme poté zobrazit přímo pomocí programu

GMSH, nebo využít uložení do různých formátů dat a takto vytvořený soubor dále použít v jiném vizualizačním programu. Je možné nadefinovat několik možností pro vykreslení výsledku. Vykreslovaná veličina je vždy přímo propojená s objektem `PostProcessing`, je možné tedy vykreslit pouze hodnoty, v ní obsažené.

Příkaz `Print` slouží k vykreslení hodnot výrazů. V našem konkrétním případě budou vykresleny hodnoty rozložení elektrostatického potenciálu na všech prvcích všech regionů (`sum`). Výraz `File "Elektrostat_pole.pos"` definuje soubor, do kterého budou uloženy vykreslené hodnoty.

```
PostOperation {
      { Name pole; NameOfPostProcessing POST_electrostatic;
            Operation {
            Print [v, OnElementsOf Sum, File "Elektrostat_pole.pos"];
                        }
      }
         }
```

`Jacobian` objekt zde reprezentuje Jacobiho determinant. Jeho vlastnosti lze využít v objektech `Formulation` a `PostProcessing`. Jedná se o algoritmus, který se využívá pro výpočty integrálních výrazů. Je propojen s objektem `Group` a charakterizuje geometrické uspořádání odpovídajících prvků (přímek, trojúhelníků, atd.).

```
Jacobian {
        { Name Vol ;
                    Case {
                    { Region All ; Jacobian Vol ; }
                         }
        }
         }
```

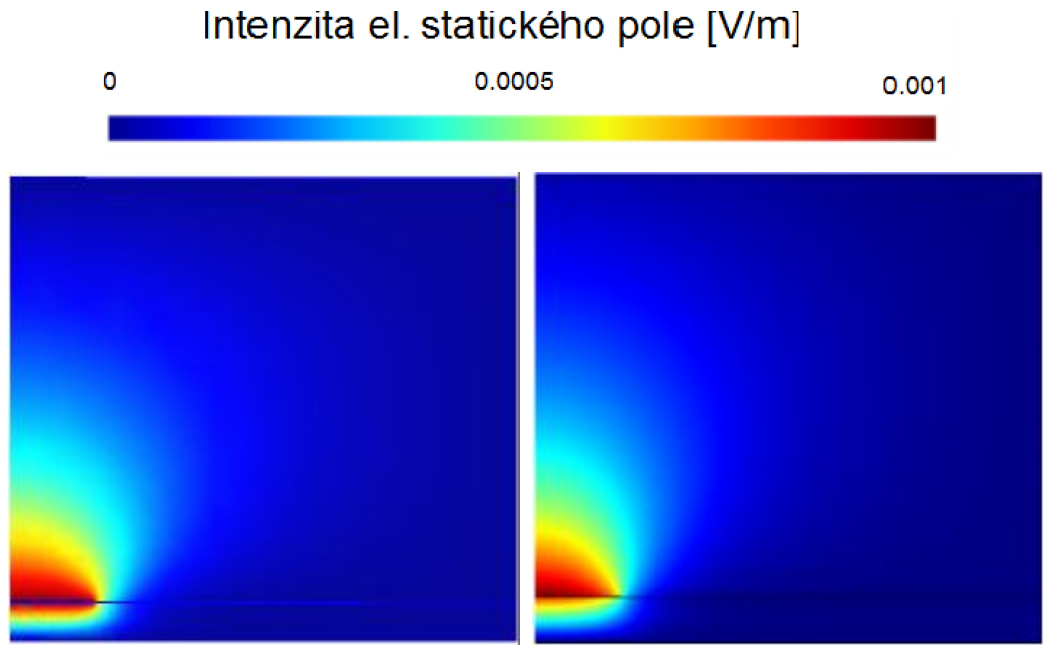
Posledním objektem je objekt `Integration`. Využívá se k výpočtu integrálních výrazů za pomoci Gaussovy numerické integrace. Princip spočívá v zavedení aproximace integrantu pomocí polynomu zvoleného stupně a následná integrace tohoto polynomu. Je tedy nutné pro každý typ prvku specifikovat počet integračních bodů. Pro popis úsečky je dostačující popis pomocí tří integračních bodů, pro popis trojúhelníku jsou nutné již minimálně čtyři integrační body.

```
Integration {

      { Name I1 ;
            Case { {Type Gauss ;
                  Case { { GeoElement Point ; NumberOfPoints  1 ;}
                        { GeoElement Line ; NumberOfPoints  3 ;}
```

```
                    { GeoElement Triangle ; NumberOfPoints  4 ;}
                    { GeoElement Quadrangle ; NumberOfPoints  4 ;}
                    { GeoElement Tetrahedron ; NumberOfPoints 4 ;}
                    { GeoElement Hexahedron ; NumberOfPoints  6 ;}
                    { GeoElement Prism ; NumberOfPoints  9 ; } }
                }
            }
               }
```

Objekty `Jacobian` a `Integration` se z důvodu přehlednosti zapisují do oddělených souborů. Jsou tedy nezávislé na zdrojovém kódu a je možné jejich požití v jakékoli jiné řešené úloze. V případě potřeby se pouze upraví počet integračních bodu jednotlivých prvků, čímž dosáhneme vyšší přesnosti výpočtu. Prodlužuje se tím však výpočetní čas. Pro přístup k těmto odděleným objektům je nutné do programu vložit příkazy:

```
Include "IntJac.pro"

Include "BlackBox_Elektostat.pro"
```



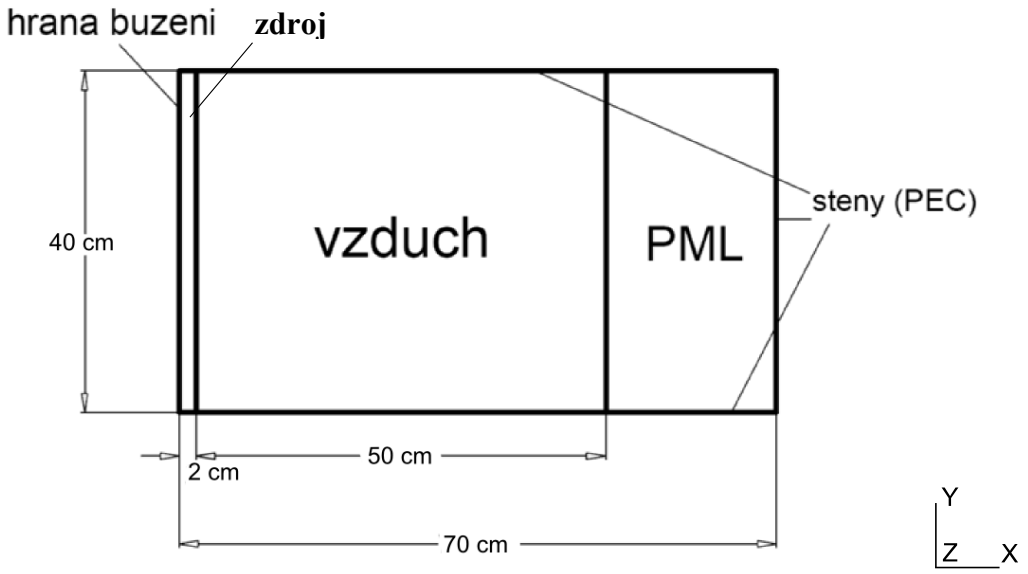
a) b)

Obr. 10 *Rozložení elektrostatického pole v okolí mikropásku: a) výsledek pomocí programu GETDP b) výsledek pomocí program COMSOL Multiphysics*

Na Obr. 10 je znázorněno rozložení elektrostatického pole v okolí mikropásku. Lze pozorovat vliv rozložení elektrostatického pole v závislosti na vlastnostech okolního prostředí, kde v oblasti dielektrika je pole postupně pohlcováno.

10.1 *Šíření elektromagnetické vlny v uzavřeném prostředí*

Tento příklad je zaměřen na šíření elektromagnetické vlny v uzavřeném prostředí. Bude zkoumáno především rozložení příčně elektrického vidu TE. Zkoumaný prostor se skládá ze tří částí (Obr. 11). V části *zdroj* je buzena elektromagnetická vlna. Tato vlna se šíří nejprve prostředím, jehož vlastnosti odpovídají vakuu. Hodnoty permitivity i permeability jsou rovny ε_0 a μ_0. Dále se vlny šíří do prostředí *PML*, které představuje absorpční prostředí. Jeho úkolem je pohltit vstupující vlnu a zabránit tak případným odrazům. Hrana *buzeni* slouží jako budící hrana vlnovodu. Ostatní hrany jsou nastaveny jako dokonalý elektrický vodič (PEC - Perfect Electric Conductor), kde složka $E_z = 0$. Oblast *PML* je definována jako prostředí s absorbčními vlastnostmi.



Obr. 11 *Geometrický tvar vlnovodu pro šíření vlny (frekvenční oblast)*

Zdrojový kód je rozdělen do čtyř různých souborů. Soubor `frekvencni_oblast.geo` reprezentuje geometrii úlohy. V souboru `IntJac.pro` jsou uloženy pomocné objekty `Jacobian` a `Integration` , sloužící pro výpočet řešení úlohy. Soubor `frekvencni_oblast.pro` obsahuje definice objektů `Group`, `Function`, `Contraint` a `PostOperation`. V posledním souboru `BlackBox_Freq.pro` se nachází objekty `FunctionSpace`, `Formulation`, `Resolution` a `PostProcessing`. Hlavním úkolem toho rozdělení je snadná následná modifikace. Pouhou úpravou soboru `frekvencni_oblast.pro` je možná modifikace parametrů řešené úlohy. Ostatní soubory můžeme označit jako „BlackBox*"* a již do jejich obsahu nezasahovat.

Postup pro vytvoření zdrojového kódu je obdobný, jako v minulé úloze. Z tohoto důvodu se zaměřím na ty části kódu, které se odlišují. Nejprve je nutné vytvořit geometrii objektu. Její rozměry jsou 70 × 40 cm, kde oblast buzení je široká 2 cm a oblast *PML* vrstvy odpovídá šířce 20 cm (Obr. 11). Využijeme k tomu program GMSH, nebo obyčejný textový editor. Princip vytváření geometrie je popsán v minulé úloze.

Nejdříve si vysvětlíme objekty ze souboru `frekvencni_oblast.pro`. V objektu `Group` je nutné přiřadit k oblastem geometrie různé proměnné, které budou danou oblast reprezentovat. Jedná se o hranu buzení, kde bude vznikat elektromagnetické vlnění. Stěny vlnovodu pro definování okrajových podmínek a také oblasti *vzduch* a *PML*. Také se zde

nachází oblasti *Vol* a *Tot*, reprezentující spojení několika regionů do větších celků. Vol sdružuje regiony *vzduch* a *PML*, které budou využity v procesu výpočtů a postprocesingu. Tot připojuje k regionu Vol také hranu *buzení*.

Objekt `Function` obsahuje definice všech konstant a také vlastnosti jednotlivých oblastí. Je nutné definovat permitivitu `epsilon`, permeabilitu `nu`, měrnou vodivost prostředí `sigma`, budící kmitočet `Freq` a vlastnosti oblastí *zdroj a vzduch.*

```
nu0 = 1 / (4.e-7 * Pi) ;
ep0 = 8.854187817e-12 ;
sigma   [ Vol ]   = 0 ;
Freq = 6.0e8;
epsilon [ Region[{zdroj,vzduch}] ]   = ep0 ;
nu      [ Region[{zdroj,vzduch}] ]   = nu0 ;
```

Dále je nutné definovat oblast *PML,* neboli dokonale přizpůsobenou oblast (PML - Perfect Matched Layers). Využívá se při simulacích šíření elektromagnetických vln na hranicích simulačních oblastí. Jejím cílem je zamezit vzniku interferencí odražených vln s vlnou přímou. Tato vrstva je modelována jako ztrátový materiál s dvourozměrnou strukturou, který vykazuje anizotropii v jednom směru. Přímo popis PML vrstvy je velmi komplikovaný, proto se jako náhrada často využívají absorpční vrstvy.

```
c[] = Complex[1,-1*(X[]-0.4)] ;
tens[] = TensorDiag[1/c[],c[],c[]];
epsilon [ pml ]   = ep0 * tens[] ;
nu      [ pml ]   = nu0 / tens[] ;
```

Uvedený kód definuje absorpční vlastnosti PML vrstvy. První řádek znamená, že se do proměnné `c[]` uloží komplexní číslo, jehož imaginární složka je závislá na aktuální souřadnici v prostoru (`X[]`) ve směru osy x. V dalším kroku je vytvořen tenzor, v jehož hlavní diagonále se nacházejí prvky proměnné *c*. Poté jsou vytvořeny tenzory relativní permitivity a permeability v oblasti *PML*, kdy maximální absorpce vlny je ve směru osy *x* [17].

```
buz [ buzeni ] = Vector[0,0,1] ;
```

Tímto zápisem je možné nadefinovat na budící hraně budící impuls elektromagnetické vlny. Buzení je provedeno pomocí vektoru, jehož složka *z* má hodnotu rovnou jedné.

V objektu `Constraint` je nutné definovat okrajové podmínky pro ohraničení vlnovodu. V této úloze je na všech hranách, kromě hrany budící, nastavená okrajová podmínka PEC. Složka vektoru elektrické intenzity, která je tečná k dokonalému elektrickému vodiči, musí být v tomto vodiči nulová, tedy $E_z = 0$. Tuto podmínku lze zapsat následovně

```
Constraint {
      { Name PEC ;
            Case {
                  { Region steny ; Value 0. ; }
                   }
      }
           }
```

Tento soubor bude pří simulaci úlohy brán jako hlavní část programu, proto součástí jeho kódu musí být odkazy na pomocné soubory.

```
Include "IntJac.pro"

Include "BlackBox_Freq.pro"
```

Posledním objektem v tomto souboru je objekt `PostOperation`. Jeho cílem je stejně, jako v předchozí úloze, vytvořit soubory s požadovanými daty, která lze využít k zobrazení výsledků.

```
PostOperation {
      { Name e ; NameOfPostProcessing MW_e_2D ;
            Operation {
                  Print[ pole, OnElementsOf Vol , File "pole.pos" ];
                  Print[ norm, OnElementsOf Vol , File "norm.pos" ];
                  Print[ vektory, OnPlane {{0,0,0}{0.7,0,0}{0,0.4,0}}
                        {100,50} , File "vektory.pos" ] ;
                 }
       }
            }
```

V souboru `BlackBox_Freq.pro` nalezneme objekty, které není nutné při modifikaci parametrů úlohy měnit. Prvním objektem je `FunctionSpace`, který je definován typem interpolačního pole, hlavní funkcí a volitelnými okrajovými podmínkami. `Type Form1P` se využívá ke zkoumání šíření elektromagnetické vlny v kolmém řezu vlnovodu. Pomocí `Constrains` je do `FunctionSpace` vztažena okrajová podmínka, definovaná výše.

```
FunctionSpace {

      { Name Hcurl_2D; Type Form1P;
         BasisFunction {
            { Name sn; NameOfCoef en; Function BF_PerpendicularEdge;
              Support Tot; Entity NodesOf[All]; }
                       }
         Constraint {
            { NameOfCoef en; EntityType NodesOf ; NameOfConstraint
              PEC; }
                  }
          }
```

V objektu `Formulation` je využita Galerkinova metoda. Využívá postup používaný při řešení soustavy parciálních diferenciálních rovnic, jehož princip spočívá v nahrazení původní rovnice, tzv. silné formulace, její integrální formou, tzv. slabým řešením, a následnou diskretizací slabého řešení. Převedení rovnice do tzv. slabé formulace je podrobně popsáno v literatuře [22]. V této úloze budeme řešit vektorovou vlnovou rovnici elektrické složky pole *Ez*.

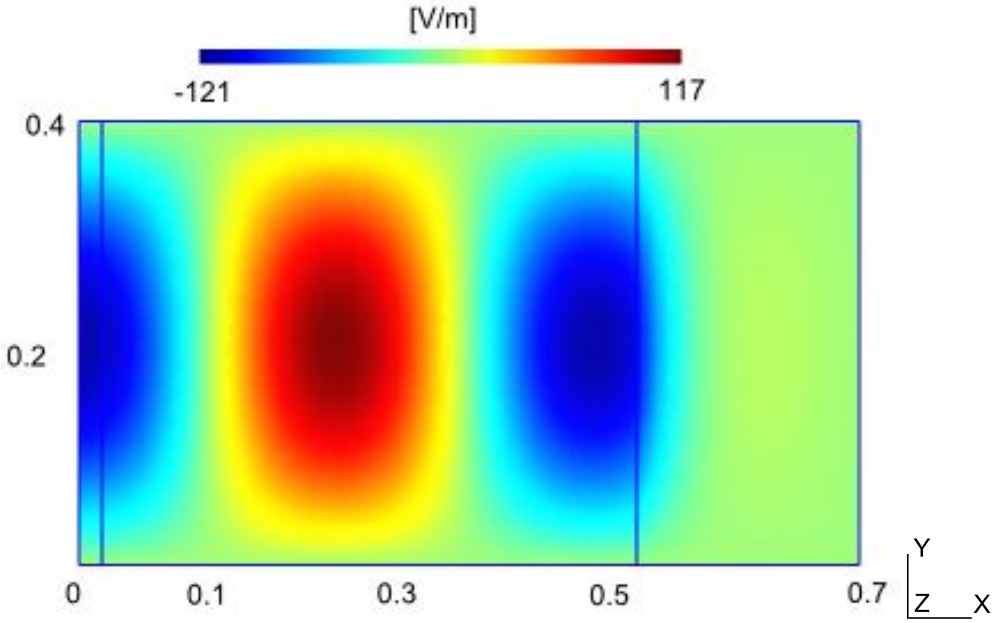
```
Formulation {
     { Name FORvlnovod_2D; Type FemEquation;

       Quantity {
         { Name e; Type Local; NameOfSpace Hcurl_2D; }
       }
       Equation {
           Galerkin { DtDt [ epsilon[] * Dof{e} , {e} ];
                      In Vol; Integration I1; Jacobian JVol;  }
           Galerkin { [ nu[] * Dof{Curl e} , {Curl e} ];
                      In Vol; Integration I1; Jacobian JVol;  }
           Galerkin { [ - buz [] , {e} ];
                      In Sur; Integration I1; Jacobian JSur;  }
                }
     }
                }
```

V objektu `Resolution` se opět definuje, zdali se bude úloha řešit ve frekvenční, nebo časové oblasti. Zde je ukázka řešení pro frekvenční oblast.

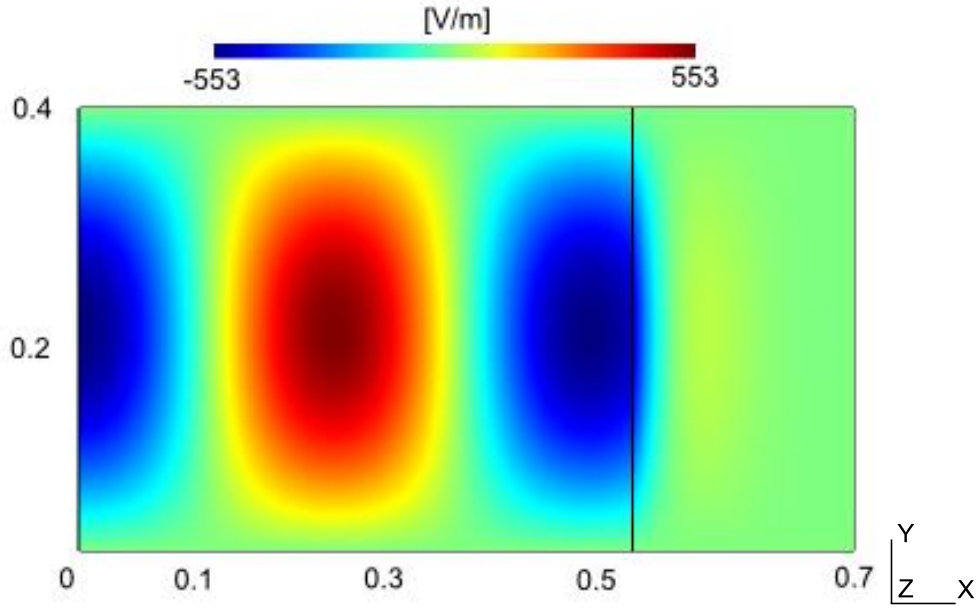
```
     Resolution {
                 { Name vlnovod_2D_FREQ;
                       System {
                             { Name A; NameOfFormulation FORvlnovod_2D;
                               Type ComplexValue; Frequency Freq; }
                               }
                       Operation { Generate[A];Solve[A]; SaveSolution[A];}
                 }
                }
```

Posledním objektem je opět `PostProcessing`, umožňující pomocí vypočítané veličiny zobrazit požadovaný výsledek analýzy pro šíření složky vlny E_z. (Obr. 12 a Obr. 13). Vykreslený výsledek *norm* vznikne po vynásobení vektoru *e* komplexně sdruženým vektorem. Jeho výsledkem je tedy bezfázové rozložení pole v dané oblasti. Pokud vykreslíme proměnnou *vektory,* získáme rozložení pole podle fázorů. Je tedy možné zobrazit pole pro různou fázi.

```
   PostProcessing {
           { Name MW_e_2D;NameOfFormulation FORvlnovod_2D;NameOfSystem A;
                 Quantity {
                       { Name pole;  Value{ Local{ [ CompZ[{e}] ] ; In
                             Tot; Jacobian JVol; } } }
                       { Name vektory;  Value{ Local{ [{e} ] ; In Vol;
                             Jacobian JVol; } } }
                       { Name norm; Value{ Local{ [ Norm[{e}] ] ; In Vol;
                             Jacobian JVol; } } }
                          }
           }
                  }
```

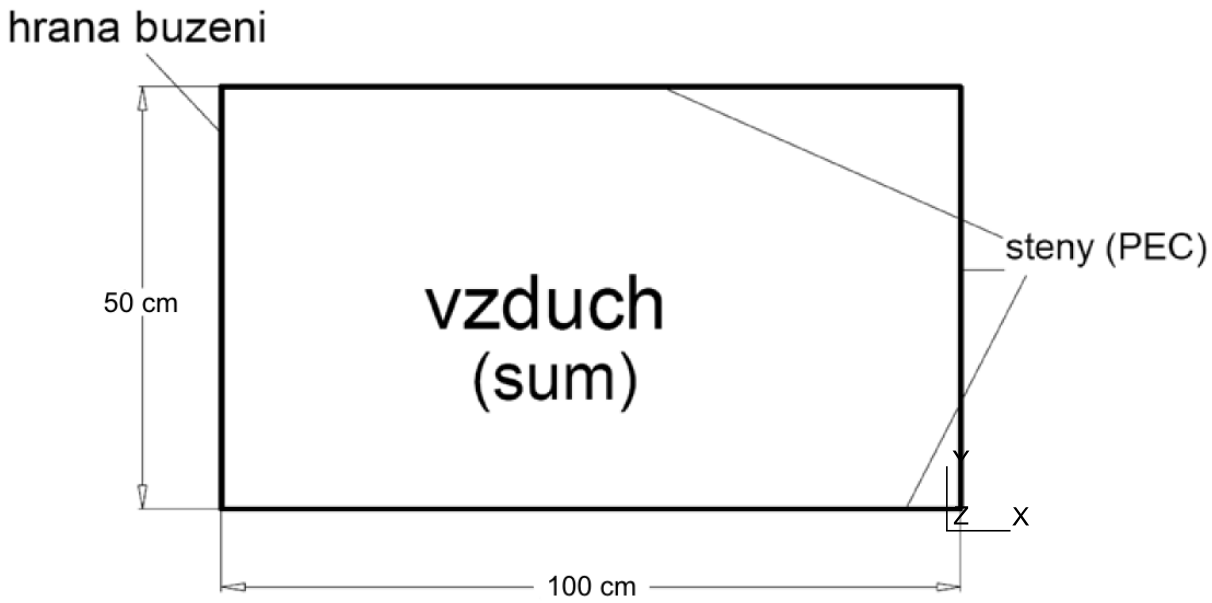


Obr. 12 *Šíření a absorpce vlny pomocí programu GETDP (při f = 750MHz).*



Obr. 13 *Šíření a absorpce vlny pomocí programu COMSOL (při f = 750MHz).*

Při přechodu z frekvenční oblasti do časové je nutná změna zdrojového kódu v objektech `Function`, `Constrains` a `Resolution`. Budou zde tedy uvedeny pouze rozdíly zdrojového kódu oproti řešení úlohy ve frekvenční oblasti. Rozměry vlnovodu pro časovou oblast byly zvoleny dle Obr. 14. Řešená úloha je součástí příkladu počítačových cvičení č. 1 [25]. Oblast PML v časové oblasti pro aktuální verzi programu GETDP je velice obtížné vytvořit. Vyžadovalo by to náročnou matematickou formulaci dané PML oblasti. PML pro časovou oblast tedy není zahrnuta. Celý vlnovod se tedy skládá z jedné oblasti se stejnými vlastnostmi (oblast `sum`).



Obr. 14 *Geometrický tvar vlnovodu pro šíření vlny (časová oblast)*

Objekt `Function` obsahuje definice permitivity `epsilon` a permeability `nu` pro vakuum. Tyto vlastnosti jsou dále přiřazeny oblasti `sum`, tedy celému vlnovodu. Dále jsou definovány parametry pro řešení úlohy v časové oblasti. Jedná se o počáteční čas simulace, časový krok, konečný čas simulace, parametry beta a gamma. Nastavením těchto parametrů lze zkoumat šíření vlny vlnovodem v určitých časových intervalech. Funkce `TimeFct[]` definuje budící puls, který je charakterizován amplitudou `A` a dobou náběhu `tr`. Tento tvar pulsu je využíván pro definici časově proměnných proudů, které jsou příčinou vzniku elektromagnetického pole s vektorem elektrické intenzity kolmým na nákresnu [18].

```
nu      [ sum ] = nu0 ;
epsilon [ sum ] = ep0;

t_min = 0;
t_max = 4.0e-9;
dt    = 0.02e-9;
beta = 0.25;
gamma = 0.5;

A = 1;
ny = 2;
tr = 0.5e-9;
alfa = ny/tr;
TimeFct[] = A*((alfa*$Time/ny)^ny*Exp[-alfa*$Time+ny])*($Time>0);
```

Časový krok `dt` je vhodné nastavit jako malý zlomek celkového času. Tím se vyhneme nesprávnému vykreslení časového průběhu. K němu dochází v případě rychlé změny signálu, která je kratší, nežli délka časového kroku analýzy. Nevýhodou je však prodloužení doby výpočtu.

Na stěnách vlnovodu jsou nastaveny okrajové podmínky PMC. Nulová hodnota tečné složky magnetického pole je součásti integrální formulace problému Garlekinova integrálu. Pokud tedy nepřiřadíme okrajovým stěnám v objektu `Constrains` nulovou hodnotu, bude platit na hranicích $n \times H = 0$. Bude se tedy jednat o okrajovou podmínku PMC. Buzení elektromagnetické vlny je zajištěno na hraně `buzeni` pomocí časově závislé funkce `TimeFct[]`, která je definována výše.

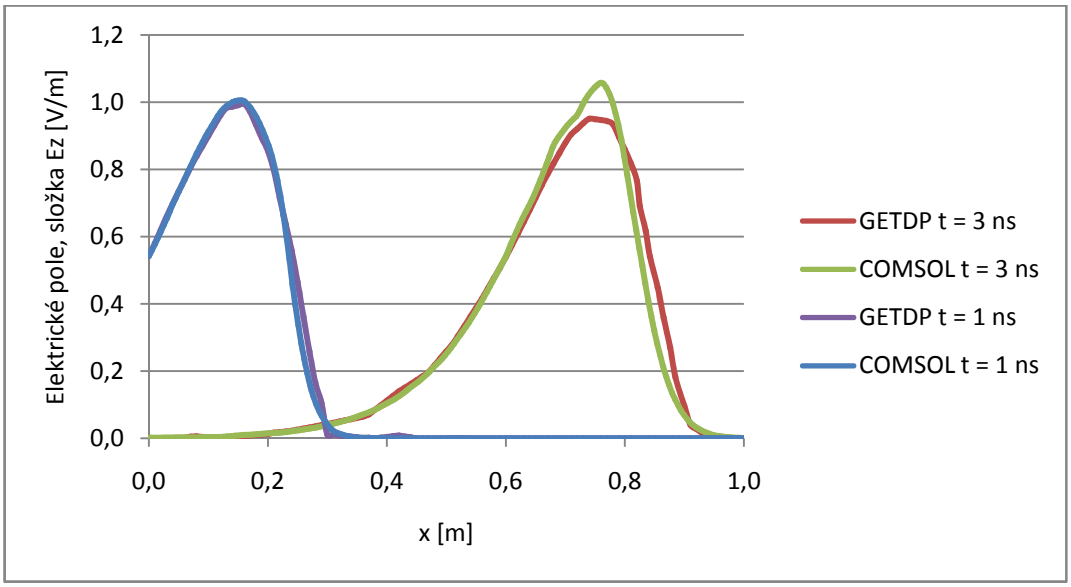
```
Constraint {
  { Name BC ;
    Case {
        { Region buzeni ; Value 1.0; TimeFunction TimeFct[]; }
    }
  }
}
```

Také je pro simulace v časové oblasti nutné změnit objekt `Resolution`. Využívá se zde aproximace založené na Newmarkově metodě. Tato metoda je velice efektivní při použití pro lineární úlohy, kde není nutné měnit velikost časového kroku. Parametry této aproximace jsou výše definované proměnné `t_min`, `t_max` a `dt`. Parametry `beta` a `gamma` ovlivňují přesnost a stabilitu výsledného algoritmu.

```
Resolution {
  { Name vlnovod_2D_TIME;
        System      {
            { Name A; NameOfFormulation FORvlnovod_2D; }
                }
        Operation {
            InitSolution[A] ;
            InitSolution[A] ;
            TimeLoopNewmark[t_min,t_max,dt,beta,gamma]
{
            Generate[A] ; Solve[A] ; SaveSolution[A] ;
            }
                }
  }
        }
```

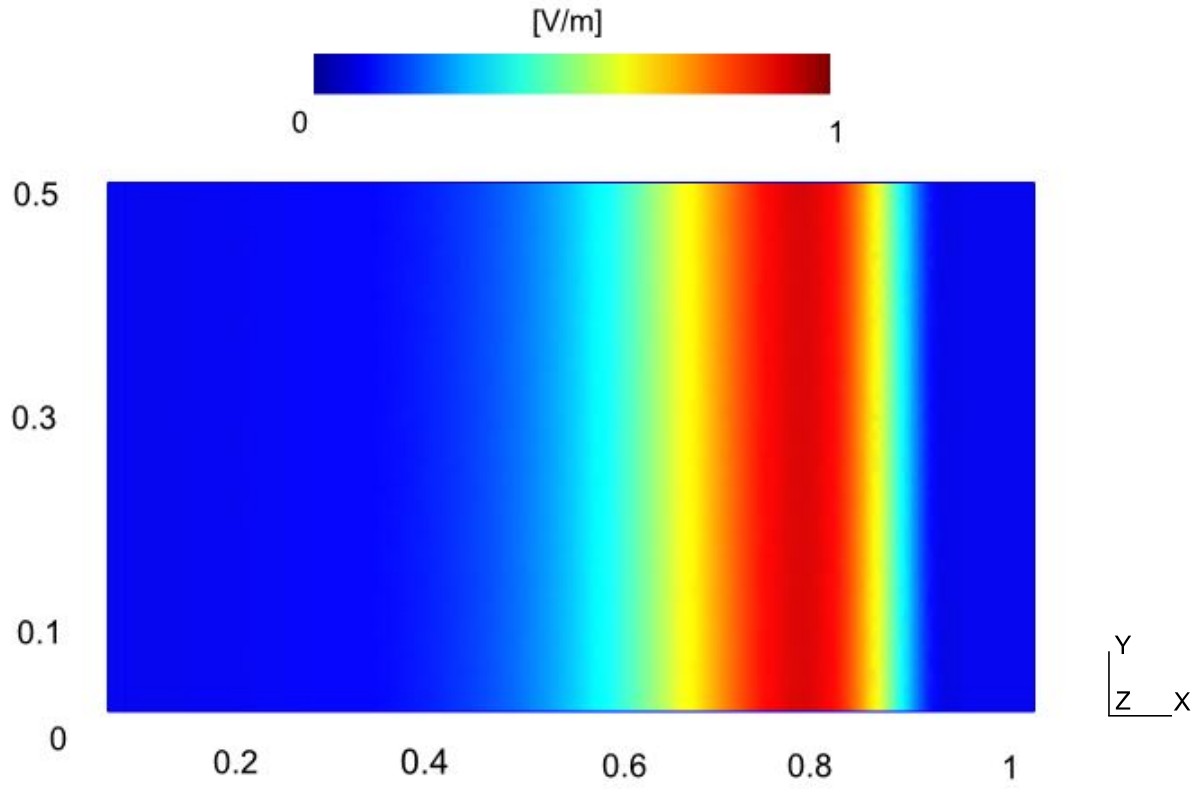
Pro potřeby exportu dat je možné v programu GETDP využít objekt `PostOperation` a příkaz `TimeTable`. Tento příkaz umožňuje vyexportovat data z dané oblasti a uložit je do textového souboru. V textovém souboru jsou data uložena do pěti sloupců, oddělených mezerou. Jednotlivé sloupce reprezentují *časový krok*, *čas* a souřadnice *x*, *y* a *z*. Příkaz `TimeStep()` udává časový krok, pro který daná data potřebujeme získat. Následující zápis vyexportuje data, nacházející se na ose souměrnosti vlnovodu.

```
Print [osa, OnLine { {0, 0.25, 0} {1, 0.25, 0} } {100},
   TimeStep{59}, File> "osa.txt", Format TimeTable];
```

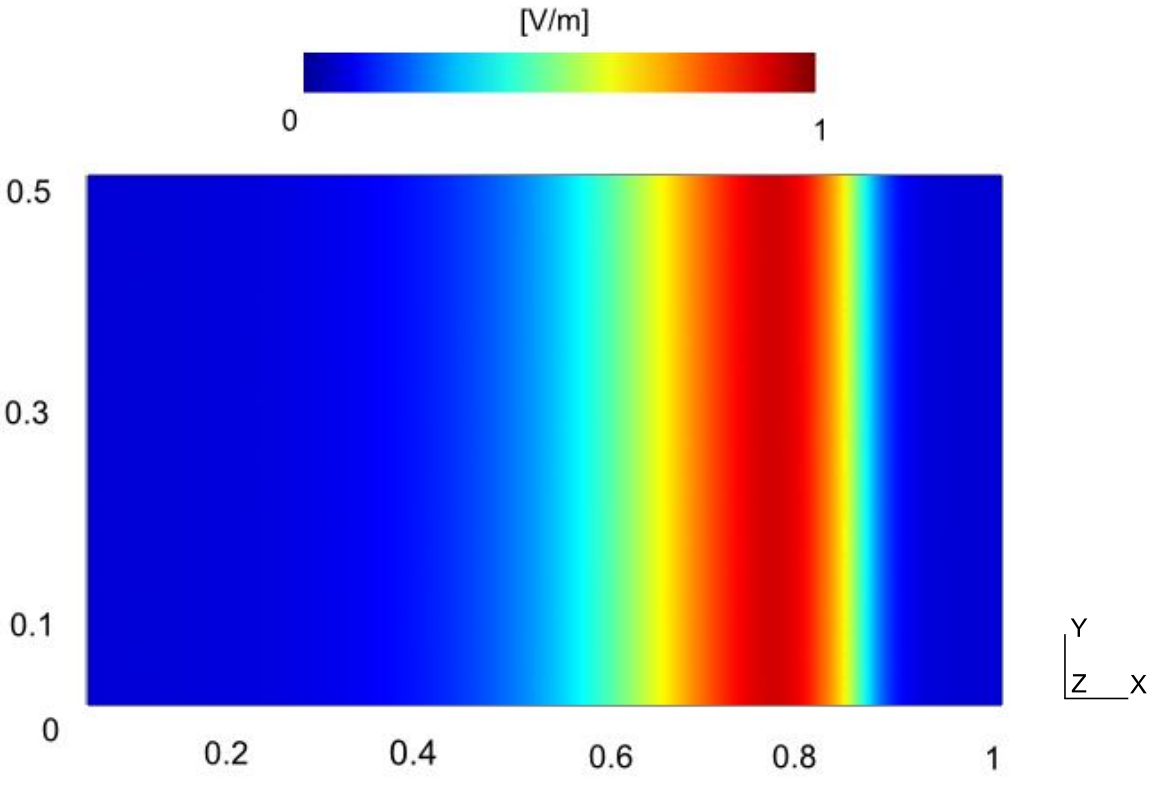


Obr. 15 *Porovnání průběhu budícího pulsu f(t) v různých časech*

Na Obr. 15 je zobrazen průběh budícího pulsu *f(t)*, kterým je vlnovod buzen. Jsou zde zobrazeny pulsy pro dva časové okamžiky. Při porovnání Obr. 15 a Obr. 17 je možné pozorovat, že maximum intenzity složky elektrického pole *Ez* v čase t = 3ns se skutečně nachází ve vzdálenosti 0,75 m od počátku souřadnic. Při exportu potřebných dat si programy GETDP a COMSOL vytváří vlastní řetězec hodnot pro osu x. Program GETDP vytváří hodnoty rovnoměrně rozložené v celé oblasti. COMSOL vytváří data dle aktuální strmosti růstu hodnot závislé proměnné. Tato skutečnost může být příčinou, že se pulsy v daných časových okamžicích plně nepřekrývají.



Obr. 16 *Šíření složky Ez elektrického pole v čase t = 3 ns pomocí GETDP.*



Obr. 17 *Šíření složky Ez elektrického pole v čase t = 3 ns pomocí COMSOL.*

Obr. 16 a Obr. 17 znázorňují rozložení elektrického pole v čase t = 3ns. Vlnovod je buzen pulsem f(t), který charakterizuje rovinnou vlnu. Pro hladký průběh šíření vlny je vhodné zvolit jemný časový krok. V opačném případě může dojít k nepatrným numerickým chybám, které by se projevily jako menší sekundární vlny za vlnou hlavní.

10.2 *Problém vlastních čísel*

Program GETDP umožňuje také řešit problém vlastních čísel a vlastních vektorů. Pokud budeme uvažovat matici $\boldsymbol{A}$ řádu n a $\boldsymbol{x}$ jako vektor o n složkách, pak vlastním číslem matice $\boldsymbol{A}$ je každé komplexní číslo λ. Pro číslo λ platí rovnice

$$\boldsymbol{A}\boldsymbol{x} = \lambda\mathbf{x}, \tag{2}$$

která má netriviální řešení ($\boldsymbol{x} \neq 0$). Takový nenulový vektor se nazývá vlastním vektorem. Rovnici (2) lze dále přepsat do tvaru

$$(\boldsymbol{A} - \lambda\boldsymbol{I})\boldsymbol{x} = 0, \tag{3}$$

kde $\boldsymbol{I}$ je jednotková matice řádu n. Soustava má netriviální řešení pouze v případě, kdy determinant matice $\boldsymbol{A} - \lambda\mathbf{I}$ je nulový. Vlastní čísla matice A jsou tedy dány řešením [24]

$$det(\boldsymbol{A} - \lambda\boldsymbol{I}) = 0. \tag{4}$$

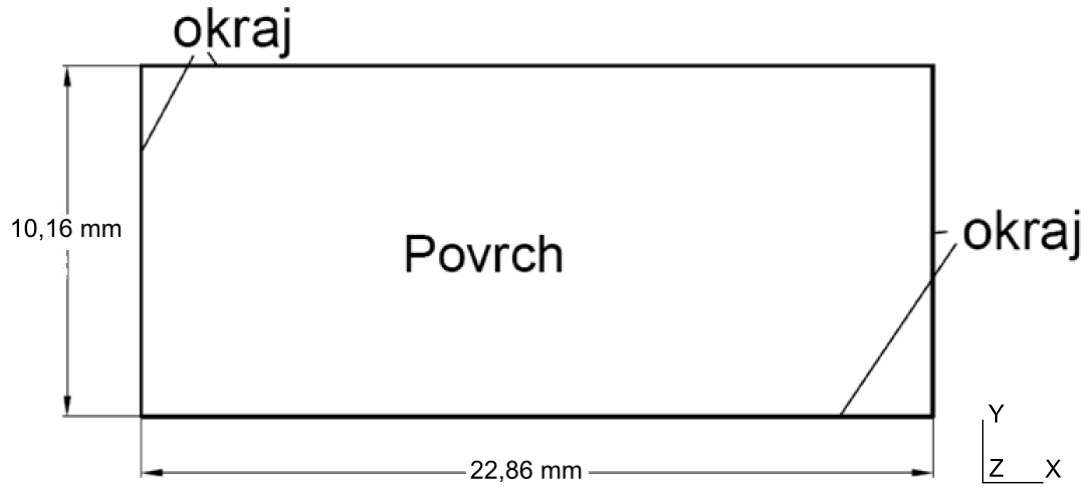
Vlastní čísla lze určit několika způsoby, jakými je výpočet pomocí charakteristického polynomu, metody využívající podobnosti matic nebo metody založené na převodu obecné matice na matici třídiagonální (Lanczosova metoda). V našem případě je využita implementovaná knihovna Arpack [20], využívající implicitně restartovaný Arnoldiho algoritmus a Lanszosovu metodu.

Řešení vlastních čísel lze využít také k určení vidů, šířících se vlnovodem. Tuto skutečnost si ověříme na úloze [26] z počítačových cvičení předmětu Elektromagnetické vlny, vedení a antény. Analýzu provedeme na jednoduchém obdélníkovém vlnovodu R100 o rozměrech 22,86 × 10,16 mm (Obr. 18). Každý vid je charakterizován dvěma přirozenými čísly m a n, která se nazývají vidová čísla. Vidy ve vlnovodu se od sebe liší svým mezním kmitočtem, fázovou a skupinovou rychlostí, útlumem atd. Pro praxi je nejžádanější vid s nejnižším mezním kmitočtem, tzv. dominantní vid. Často se využívá tzv. pásma jednovidosti, které se nachází mezi dominantním videm a prvním vyšším videm. V tomto pásmu se signál šíří bez zkreslení.

Hodnoty frekvencí prvních čtyř vidů pro TE i TM, šířících se vlnovodem, dále vypočítáme také pomocí programu COMSOL a hodnoty z obou programů porovnáme s analytickým výpočtem (5).

$$\mathrm{f_{mez}} = \frac{\frac{1}{2\pi}\sqrt{\left(\frac{m\pi}{a}\right)^2+\left(\frac{n\pi}{b}\right)^2}}{\sqrt{\varepsilon_0\mu_0}}, \tag{5}$$

kde m, n jsou přirozená čísla, charakterizující daný vid. Znaky a a b jsou rozměry vlnovodu, kde a označuje delší stranu vlnovodu. Znaky ε_0 a μ_0 označují permitivitu a permeabilitu vakua.



Obr. 18 *Geometrický tvar vlnovodu R100*

Soubor s popisem geometrie objektu musí obsahovat region pro popis celkové plochy vlnovodu a také region, na který budou aplikovány okrajové podmínky.

V objektu `Function` je nutné definovat pouze rychlost vlny. Nyní uvažujeme ideální prostředí, jehož vlastnosti odpovídají vakuu. Vlna se tedy bude šířit rychlostí odpovídající rychlosti světla.

```
Function      {
              v[]=3.e8;
              }
```

Pro řešení vlastních čísel a vidů TM je nutné definovat okrajovou podmínku PEC. Využijeme tedy objektu `Constraint` a vytvoříme okrajovou podmínku, při níž je složka $E_z = 0$. Při řešení pro vidy TE bude tato okrajová podmínka odstraněna. Tím docílíme na okrajích zkoumané oblasti nulových hodnot tečné složky magnetického pole (PMC).

```
Constraint {
        { Name PEC; Type Assign ;
         Case {
              { Region okraj ; Value 0.0 ; }
                    }
        }
   }
```

Pro vytvoření bázových funkcí využijeme objekt `FunctionSpace`. V tomto případě využijeme typ `Form0` pro vytvoření skalárního pole. Při řešení TM vidu je v rámci tohoto objektu nutné uplatnit vytvořené okrajové podmínky.

```
FunctionSpace {
  { Name VlastCisla; Type Form0;
    BasisFunction   {
      { Name se; NameOfCoef ae; Function BF_Node; Support
 Povrch; Entity NodesOf[All]; }
                    }
}
              }
```

Nyní vložíme do objektu `Formulation` rovnici, řešící problematiku vlastních čísel. K řešení využijeme obecnou formulaci vlnové rovnice.

$$\frac{\partial^2 e}{\partial t^2} = c^2 \Delta e \tag{6}$$

, kde proměnná *c* představuje rychlost šíření vlny (uvažujeme rychlost světla), *e* odpovídá vektoru vlastních čísel a Δ reprezentuje Laplaceův operátor. V následujícím bloku je tato rovnice přepsána do jazyka GETDP.

```
Formulation {
  { Name res_VlastCis; Type FemEquation;
    Quantity {
      { Name e; Type Local; NameOfSpace VlastCisla;}
               }
   Equation {
Galerkin {[ v[]*v[]*Dof{ d e},{ d e} ]; In Povrch;
   Integration I1; Jacobian Jac;}
  Galerkin {  DtDt[Dof{e}, {e} ]; In Povrch; Integration I1;
        Jacobian Jac;}
              }
  }
}
```

Takto sestavená soustava rovnic je řešena pomocí objektu Resolution. Pro výpočet vlastních čísel se využívá příkaz `EigenSolve[A,4,0,0]`. První člen v závorce reprezentuje použitý systém (vygenerovaný pomocí příkazu `GenerateSeparate[A]`), druhý člen udává množství vlastních čísel, které si přejeme vypočítat. Pro vyřešení je použita implementovaná knihovna Arpack [20], využívající implicitně restartovaný Arnoldiho algoritmus.

```
Resolution {
   { Name reseni;
    System{
      { Name A; NameOfFormulation res_VlastCis; Type
        ComplexValue; }}
    Operation{
      GenerateSeparate[A];
      EigenSolve[A,5,0,0];
      SaveSolutions[A]; }
  }
}
```

Výsledné hodnoty vlastních čísel lze zobrazit pomocí příkazů v objektu PostOperation. Jednotlivé příkazy vypíší hodnotu vlastního čísla do konsole a dále vytvoří soubor s daty, která jsou využita pro zobrazení daného vidu v postprocessingu.

```
PostOperation {
  { Name Vykresleni; NameOfPostProcessing general;
  Operation{
        Print [ c, OnElementsOf Povrch, TimeStep{0}, File
              "eigenVector0.pos"];
        Print [ c, OnElementsOf Povrch, TimeStep{1}, File
              "eigenVector1.pos"];
        Print [ c, OnElementsOf Povrch, TimeStep{2}, File
              "eigenVector2.pos"];
        Print [ c, OnElementsOf Povrch, TimeStep{3}, File
              "eigenVector3.pos"];
        Print [ c, OnElementsOf Povrch, TimeStep{4}, File
              "eigenVector4.pos"];
        }
  }
              }
```

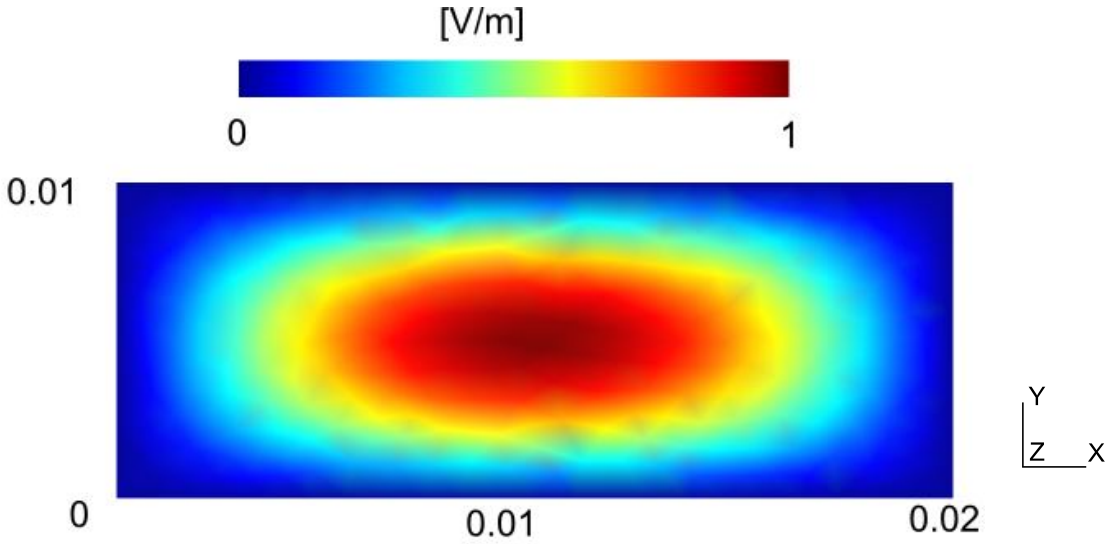
Výsledkem této simulace jsou hodnoty prvních čtyř vlastních čísel. Výsledky je možné porovnat s hodnotami získanými pomocí programu COMSOL. Také je zde uvedena hodnota získaná analytickým výpočtem dle vzorce (5). Z Tab. 4 a Tab. 5 je patrné, že se hodnoty vlastních čísel ne vždy přesně shodují. Pokud jako referenční hodnotu zvolíme analytické řešení, poté lze porovnat přesnost výpočtu obou programů. Pro vidy TE je výpočet pomocí COMSOLU velmi přesný. Hodnoty GETDP se pohybují v rozmezí do 1% od analytického řešení. Při výpočtu vidů TM dosahuje rozdíl mezi referenční a vypočítanou hodnotou u obou programů max 1,3%. Rozdíl od referenční hodnoty se zvyšuje u vyšších vidů, protože vyšší prostorové kmitočty vzorkujeme stále stejným prostorovým krokem [18].

Tab. 4 Srovnání hodnot vlastních čísel vidu TE v programech COMSOL a GETDP

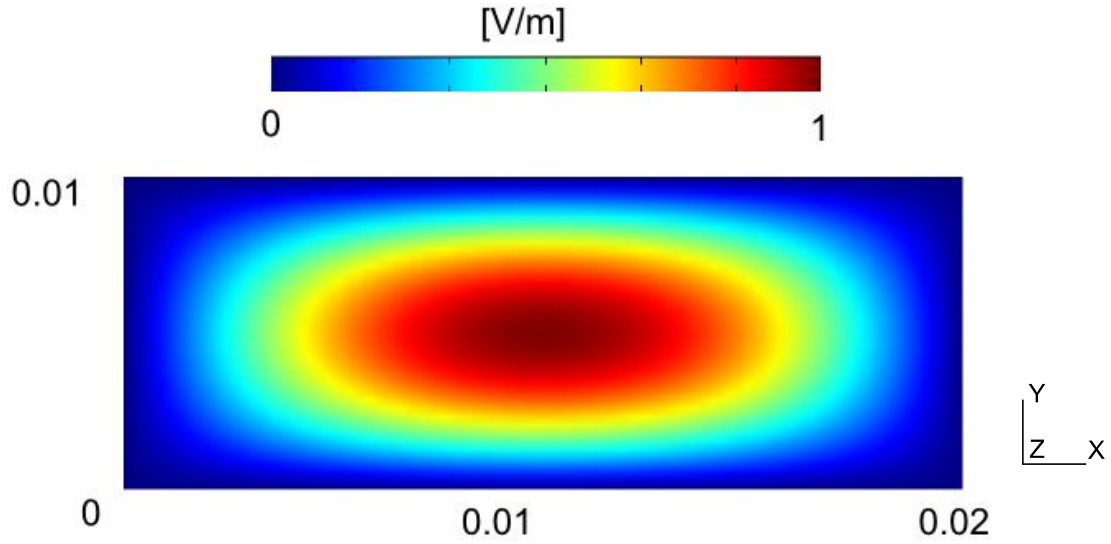
Vlnové číslo	**TE vidy**				
	GETDP [GHz]	**COMSOL [GHz]**	**Analytický výpočet [GHz]**	**Rozdíl GETDP [MHz]**	**Rozdíl COMSOL [MHz]**
1.	6,5384	6,5571	6,5583	19,9	1,2
2.	13,2425	13,1143	13,1142	-128,3	-0,1
3.	14,8355	14,7536	14,7535	-82	-0,1
4.	16,2942	16,1451	16,1450	-149,2	-0,1

Tab. 5 Srovnání hodnot vlastních čísel vidu TM v programech COMSOL a GETDP

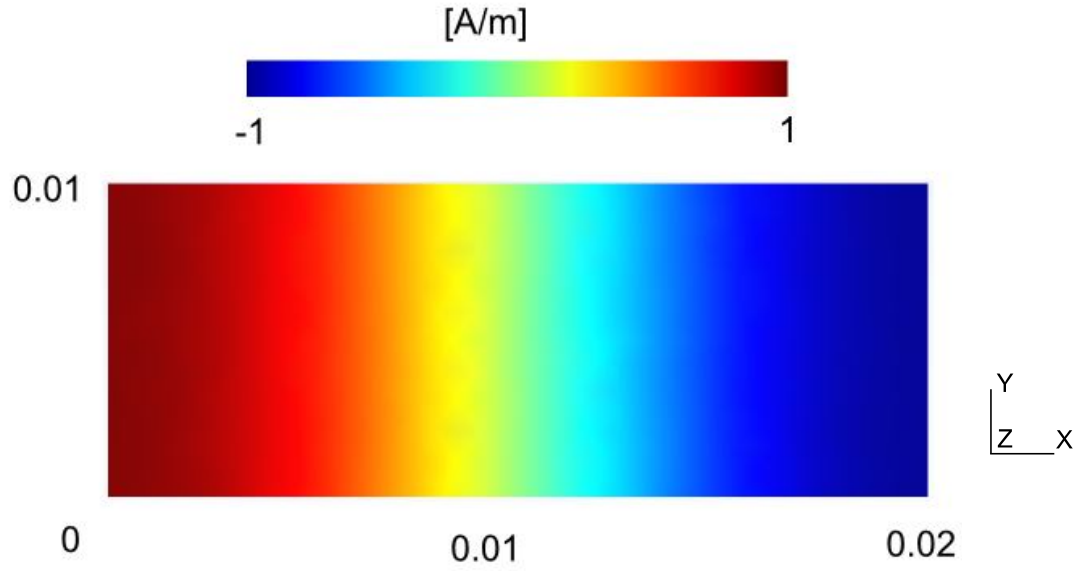
Vlnové číslo	TM vidy				
	GETDP [GHz]	**COMSOL [GHz]**	**Analytický výpočet [GHz]**	**Rozdíl GETDP [MHz]**	**Rozdíl COMSOL [MHz]**
1.	16,2109	16,1931	16,1450	-65,9	-48,1
2.	19,8420	19,8321	19,7396	-102,4	-92,5
3.	24,7819	24,7741	24,5892	-192,7	-184,9
4.	30,4793	30,4454	30,0932	-386,1	-352,2



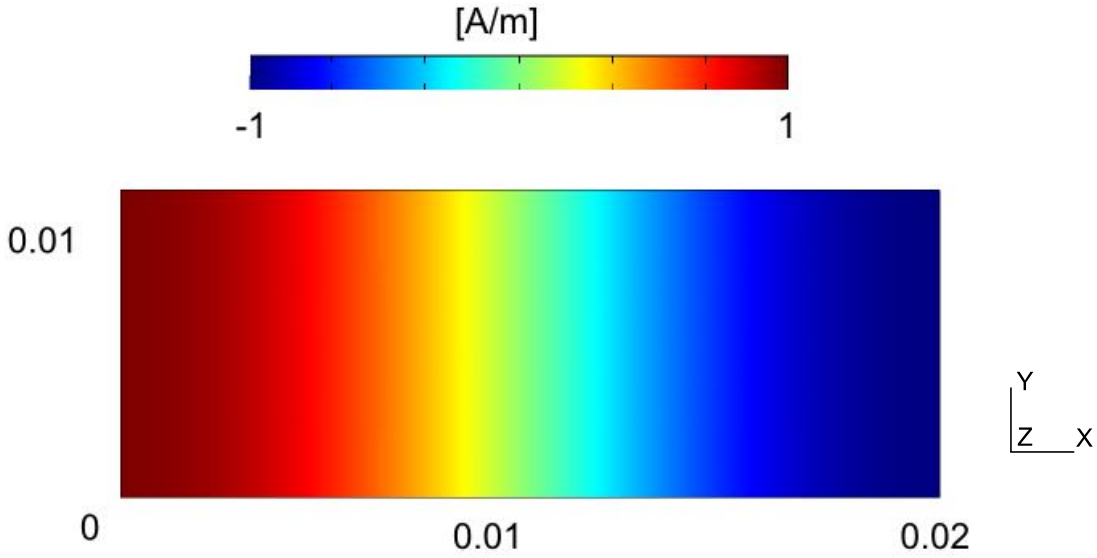
Obr. 19 *Program GETDP - ukázka vidu TM_{11} při f = 16,2109 GHz*



Obr. 20 *Program COMSOL - ukázka vidu TM_{11} při f = 16,1931 GHz*



Obr. 21 *Program GETDP - ukázka vidu TE_{10} při f* = 6,5384 *GHz*



Obr. 22 *Program COMSOL - ukázka vidu TE_{10} při f* = 6,5571 *GHz*

Dále je uvedeno na Obr. 19 až Obr. 22 zobrazení vidů TE_{10} a TM_{11}. Také zde je možné porovnat nepatrný rozdíl rozložení vidů, zobrazených pomocí programu GETDP a programu COMSOL. Příčinou může být dokonalejší vizuální zpracování výsledků pomocí programu COMSOL (vyhlazení přechodů mezi jednotlivými vykreslenými elementy). Také každý program využívá nepatrně odlišnou stupnici intenzity elektrického pole. Celková představa o průběhu vidu je však zachována.

10.3 *Difrakce na rovinné a dielektrické překážce*

Pokud je elektromagnetickému vlnění vložena do cesty překážka, je vlnění překážkou ovlivněno. V okolí této překážky lze nalézt místa, ve kterých je intenzita pole větší, nežli tomu je v prostoru bez překážky. V těchto místech dochází ke skládání přímé a odražené vlny se stejnou fází. Pokud se setká přímá a odražená vlna s opačnou fází, nastává naopak v takovém místě velmi nízká intenzita pole. Často nastává případ, kdy se překážkou stává těleso, které vlnu zcela neodrazí ani zcela nepohltí. Toto těleso se od okolí liší především hodnotou permitivity. Ta způsobuje zpomalení rychlosti šíření elektromagnetické vlny a tím dochází opět k deformaci vlnění. Tyto jevy jsou nazývány difrakcí. Difrakční jevy lze rozdělit na Fresnelovu a Fraunhoferovu difrakci. Fresnelova difrakce se zabývá především intenzitou interferenčních jevů v závislosti na poloze v konečné vzdálenosti od zdroje rovinné vlny. Naopak Fraunhoferova difrakce studuje intenzitu v rovině v nekonečnu. Jedná se tedy o zjednodušený případ Fresnelovy difrakce [23].

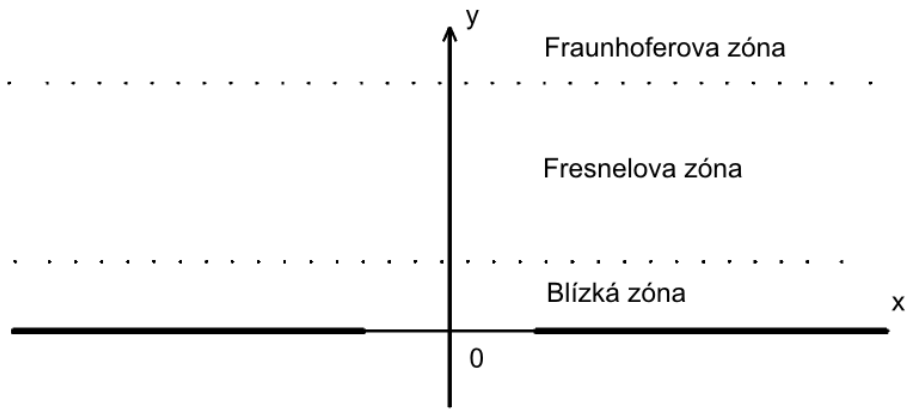
Analytický popis těchto jevů bývá velmi složitý. Využívá se Huygensova principu, podle kterého se každá část štěrbiny stává novým zdrojem vlnění. Po průchodu štěrbinou se každá vlna dále šíří všemi směry.

V této úloze se budeme zabývat difrakcí na překážce, která je ozářená z jedné strany rovinnou vlnou. V našem případě se jedná o lineárně polarizovanou vlnu ve směru osy *y*, která dopadá na překážku v rovině *xz*. Dále budeme zkoumat rozložení vlny za překážkou[21].

Pokud budeme uvažovat, že uspořádání z Obr. 23 je nezávislé na souřadnici *z*, je možné intenzitu elektrického pole za překážkou určit dle Huygensova principu dle vztahu

$$\mathrm{E}(x,y) = \mathrm{e}_z \frac{k}{2} \int_{x'=-\infty}^{\infty} E(x')\, H_0^{(2)}(kr)\, dx', \tag{6}$$

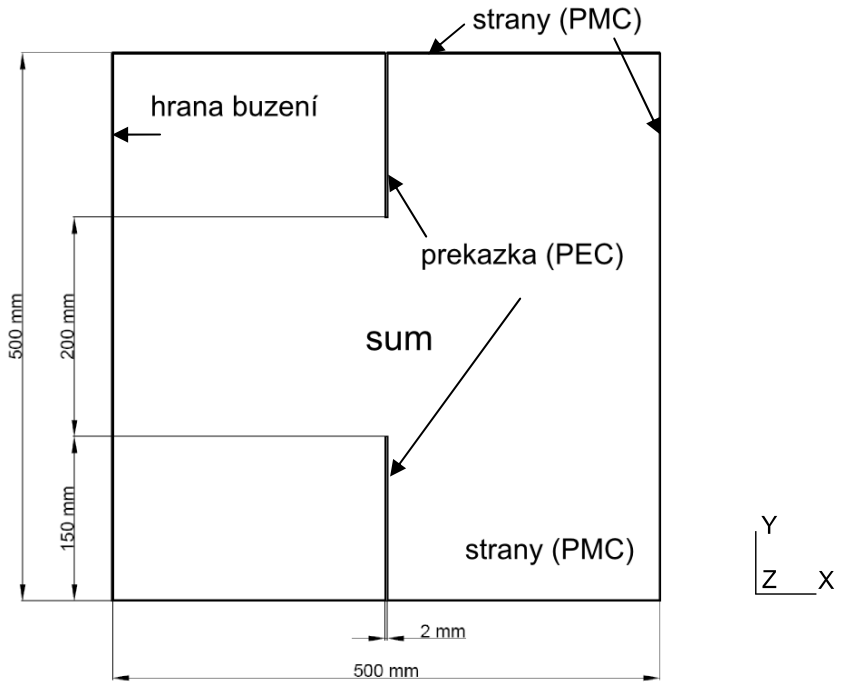
kde $E(x')$ označuje rozložení elektrické intenzity a $r = [(x-x')^2 + y^2]^{1/2}$.



Obr. 23 *Rovinná vlna dopadající na překážku s otvorem [21]*

Nyní si vytvoříme zdrojový kód pro simulaci rozložení elektrického pole *Ez* při šíření na štěrbině. Nyní vycházíme z úlohy počítačových cvičení č. 4 [27] Analýza bude provedena pro časovou oblast. Pomocí programu gmsh nebo textového editoru si vytvoříme soubor s geometrií tělesa. Vytvoříme čtverec o rozměrech 0,5 × 0,5 m. Uprostřed tohoto čtverce vytvoříme štěrbinu o velikosti 0,2 m (Obr. 24). Pro možnost porovnání bude také vytvořena štěrbina o velikosti 0,02 m. Při řešení v programu COMSOL lze překážku vytvořit pouhou

úsečkou a na tuto úsečku aplikovat okrajové podmínky. Aby však program GETDP správně rozpoznal štěrbinu a aby bylo možné na překážku uplatnit požadované okrajové podmínky, musí mít překážka nenulové rozměry. Okrajové podmínky na překážce nastavíme jako PEC, na ostatních hranách kromě hrany budící je nastavena okrajová podmínka PMC.



Obr. 24 *Geometrie vlnovodu se štěrbinou*

Soubor s geometrií bude obsahovat geometrické uspořádání dle Obr. 24 a také odpovídající regiony, použité v objektu `GROUP`.

Objekt `GROUP` obsahuje region `buzeni`, který sloužící jako budící hrana vlnovodu. Štěrbina je definována jako oblast mezi dvěma úsečkami. Tyto dvě úsečky jsou seskupeny do regionu `prekazka`. Cela oblast, na které vlnovod analyzujeme, je začleněna v regionu `sum`.

```
Group {
  buzeni   = Region[15] ;
  sum   = Region[999] ;
  prekazka = Region [16];
}
```

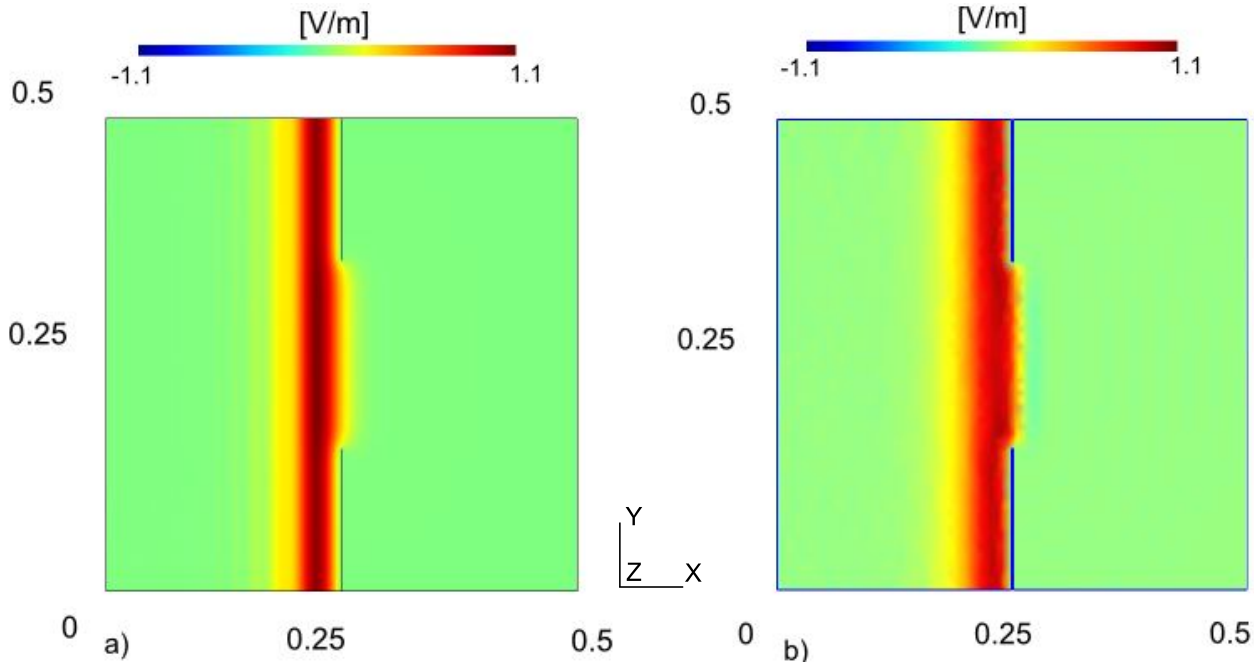
Objekt `Function` je totožný s úlohou pro analýzu vlnovodu v časové oblasti. Pouze je použit jemnější krok, aby bylo možné lépe analyzovat rozložení elektrické pole v okolí štěrbiny.

```
Function {
  dt    = 0.01e-9;
}
```

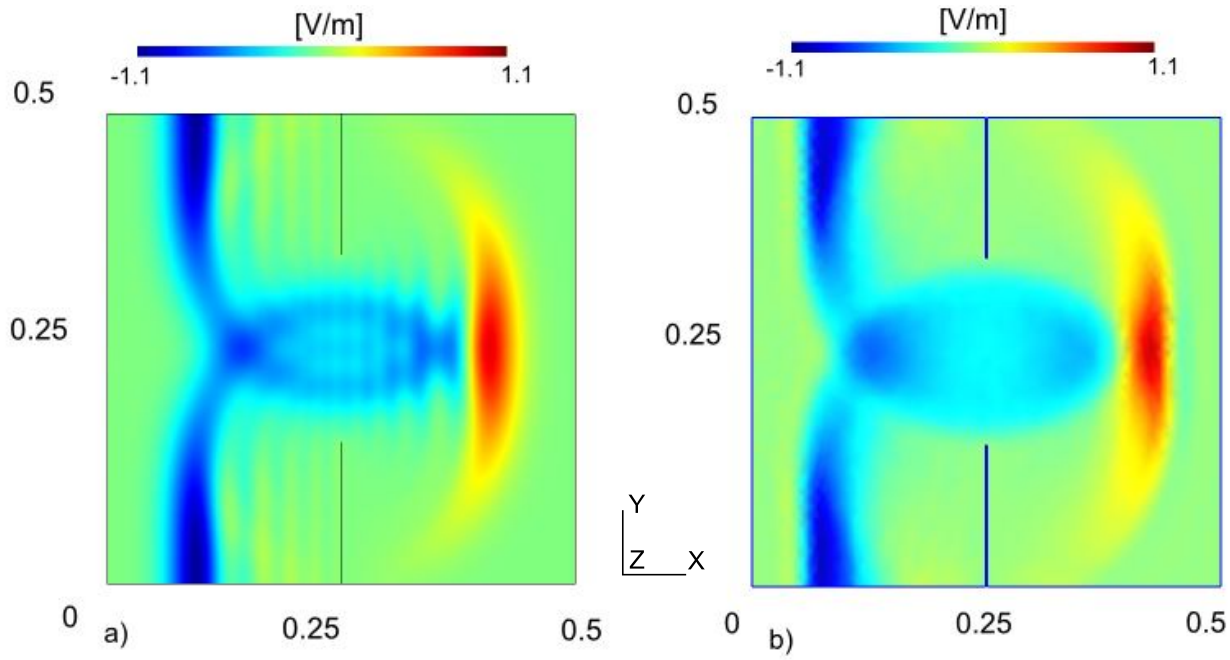
Úsečky reprezentující překážku jsou definovány okrajovou podmínkou PEC. Buzení elektromagnetické vlny je zajištěno na hraně `buzeni` pomocí časově závislé funkce `TimeFct[]`. Jako budící puls je použit puls pro definici časově proměnných proudů z úlohy 10.2. Vhledem k velikosti analyzované geometrie je však velmi dlouhý v čase. Z tohoto důvodu je změněna doba náběžné hrany pulsu na hodnotu $t_r = 0.1$ ns.

```
Constraint {
  { Name BC ;
    Case {
         { Region prekazka ; Value 0. ; }
         { Region buzeni  ; Value 1.0; TimeFunction TimeFct[]; }

    }
  }
}
```

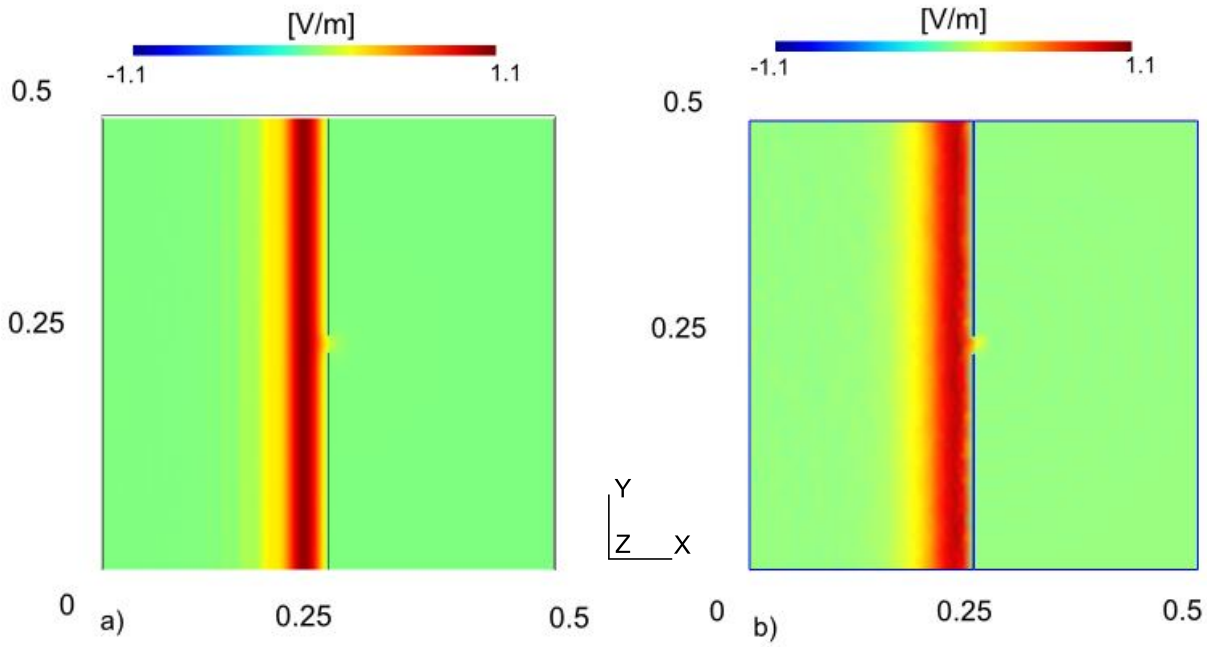
Ostatní objekty jsou součástí souboru `BlackBox.pro`. Jeho obsah však není nutné měnit. Lze využít přímo soubor, použitý pro analýzu vlnovodu v časové oblasti. Rovnice řešící danou problematiku a ostatní nástroje pro výpočty jsou vytvořeny tak, aby byly schopny bez zásahu řešit více typů úloh.



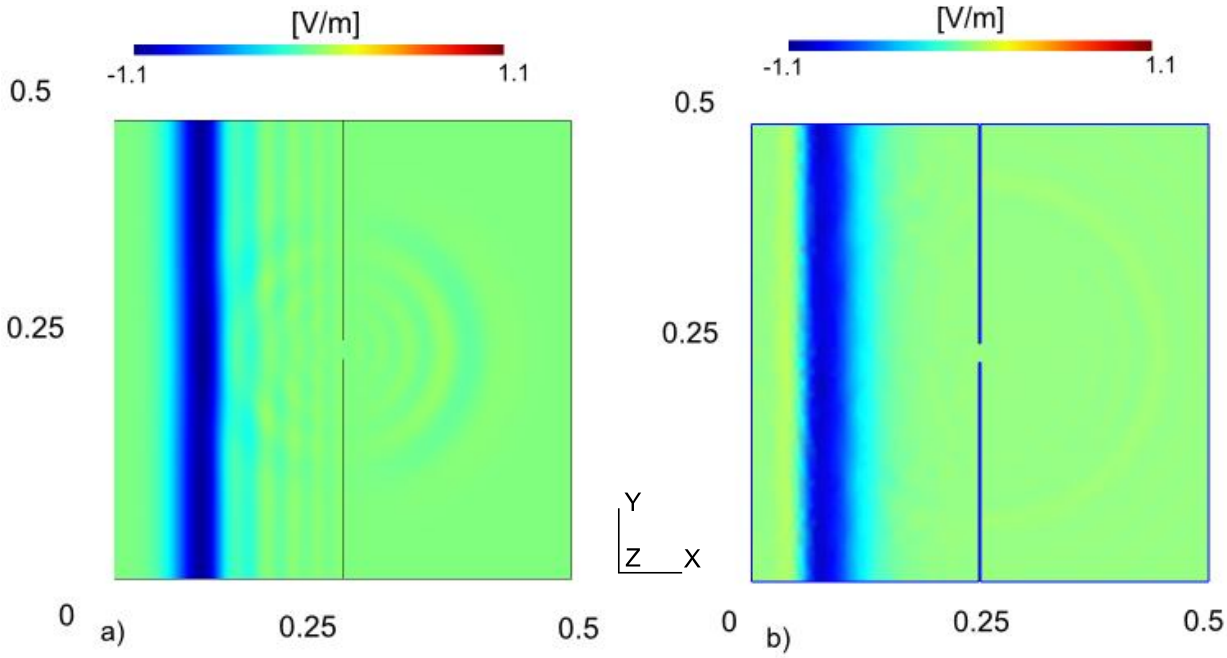
Obr. 25 *Rozložení složky Ez elektrického pole v čase t = 0.88 ns pro šířku štěrbiny 0.2 m pomocí programů a) COMSOL, b) GETDP*



Obr. 26 *Rozložení složky Ez elektrického pole v čase t = 1.5 ns pro šířku štěrbiny 0.2 m pomocí programů a) COMSOL, b) GETDP*



Obr. 27 *Rozložení složky Ez elektrického pole v čase t = 0,88 ns pro štěrbinu o šířce 0.02 m pomocí programů a) COMSOL, b) GETDP*



Obr. 28 *Rozložení složky Ez elektrického pole v čase t = 1.5 ns pro dvě štěrbiny o šířce 0.02 m pomocí programů a) COMSOL, b) GETDP*

Na Obr. 25 až Obr. 26 je zobrazeno rozložení elektrického pole složky Ez pro štěrbinu o velikosti 0,2 m. Pro názornou ukázku bylo vloženo rozložení ve dvou různých časech. Vlna se nárazem na překážku deformuje a vzniká stojaté vlnění. Vlna procházející štěrbinou se za překážkou šíří všemi směry ve tvaru vlnoploch, jejichž intenzita elektrického a magnetického pole má stejnou fázi. Tento jev je nejlépe vidět na Obr. 28. Dále byla provedena analýza na štěrbině o velikosti 0,02 m (Obr. 27 a Obr. 28). Ve všech analyzovaných časových intervalech se průběh rozložení elektrického pole shoduje s rozložením získaným pomocí komerčního programu COMSOL.

Nyní vložíme do cesty šíření elektromagnetické vlny dielektrickou překážku a budeme pozorovat změny šíření vlny v závislosti na hodnotě její permitivity. Budeme analyzovat vlnovod z Obr. 24, pouze z jeho geometrie odstraníme štěrbinu a přesně do středu vlnovodu vložíme kruh o průměru 0,2 m. Tento kruh bude representovat dielektrickou překážku, na které budeme pozorovat deformaci elektromagnetické vlny. Celý postup budeme opakovat i pro kruh o poloměru 0,02 m.

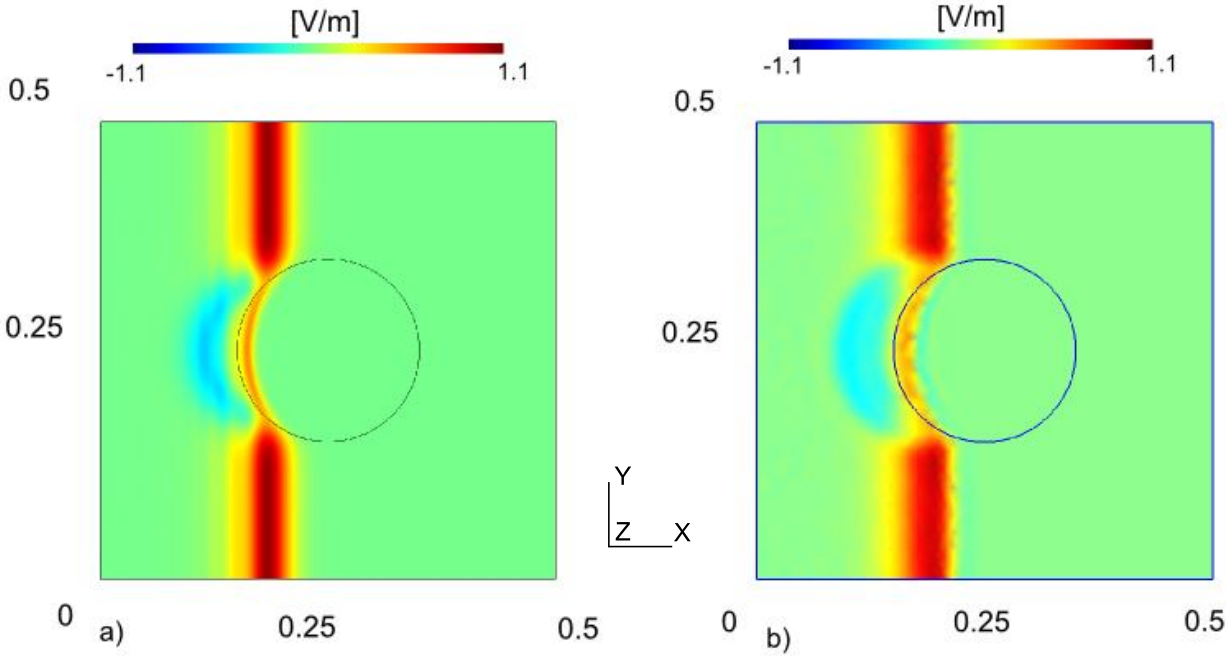
Do objektu `Group` je nutné vložit nově vytvořený region `kruh`, který representuje dielektrickou překážku. Aby bylo možné regionu `kruh` nadefinovat vlastní hodnoty permitivity, nesmí být tento region součástí celkové plochy vlnovodu.

```
Group {
        buzeni   = Region[13] ;
        kruh = Region [15];
        sum   = Region[12] ;
        sumsum   = Region[{12,15}] ;
        }
```

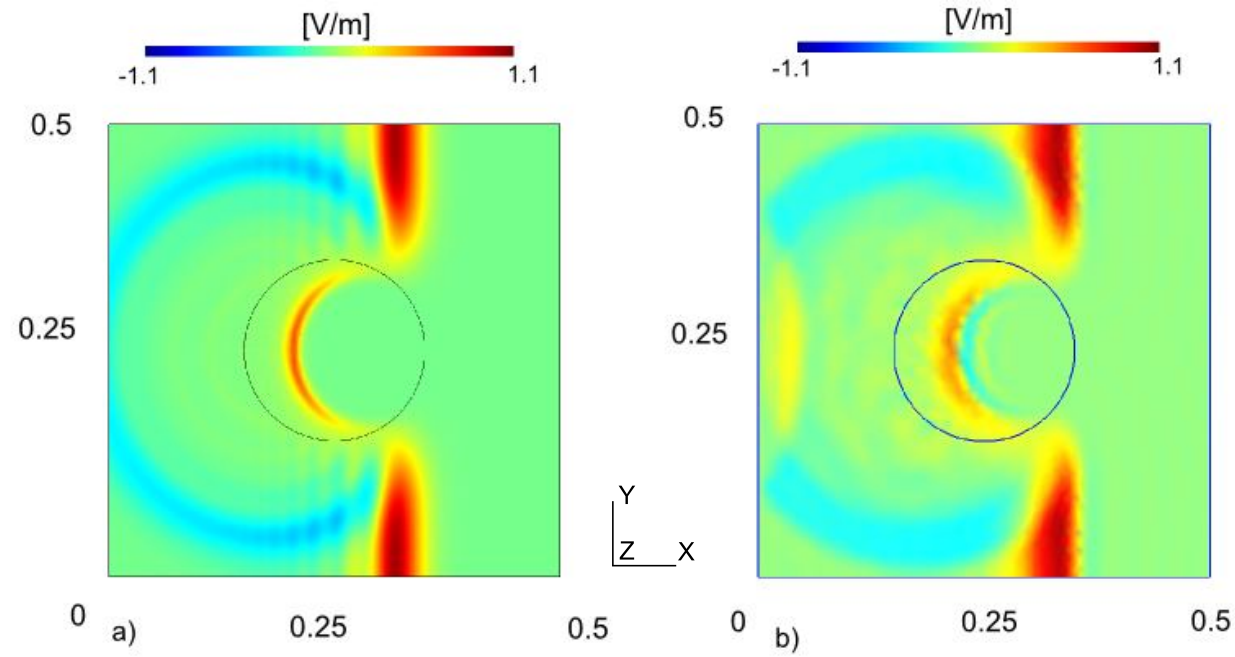
Dále je nutné přidat do objektu `Function` definici permitivity pro oblast regionu `kruh`

```
epsilon [ kruh ] = ep0*10;
```

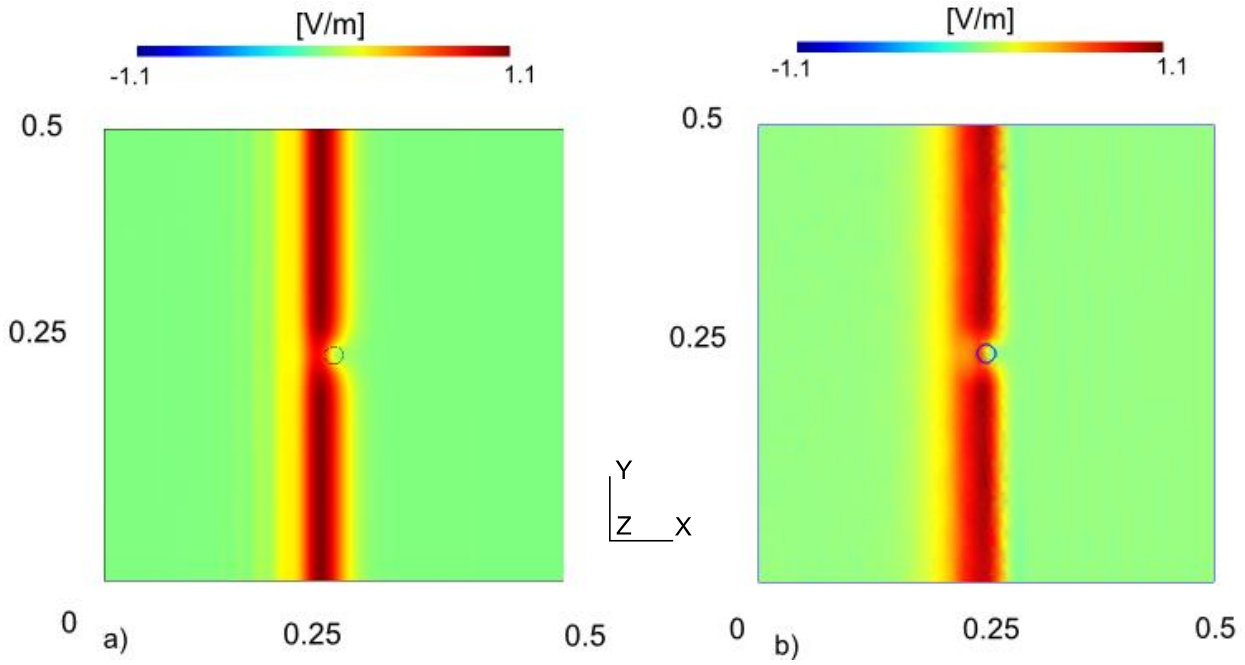
Obsah ostatních objektů zůstane zachován. Výsledky opět zobrazíme pro různé časové okamžiky a provedeme porovnání s programem COMSOL.



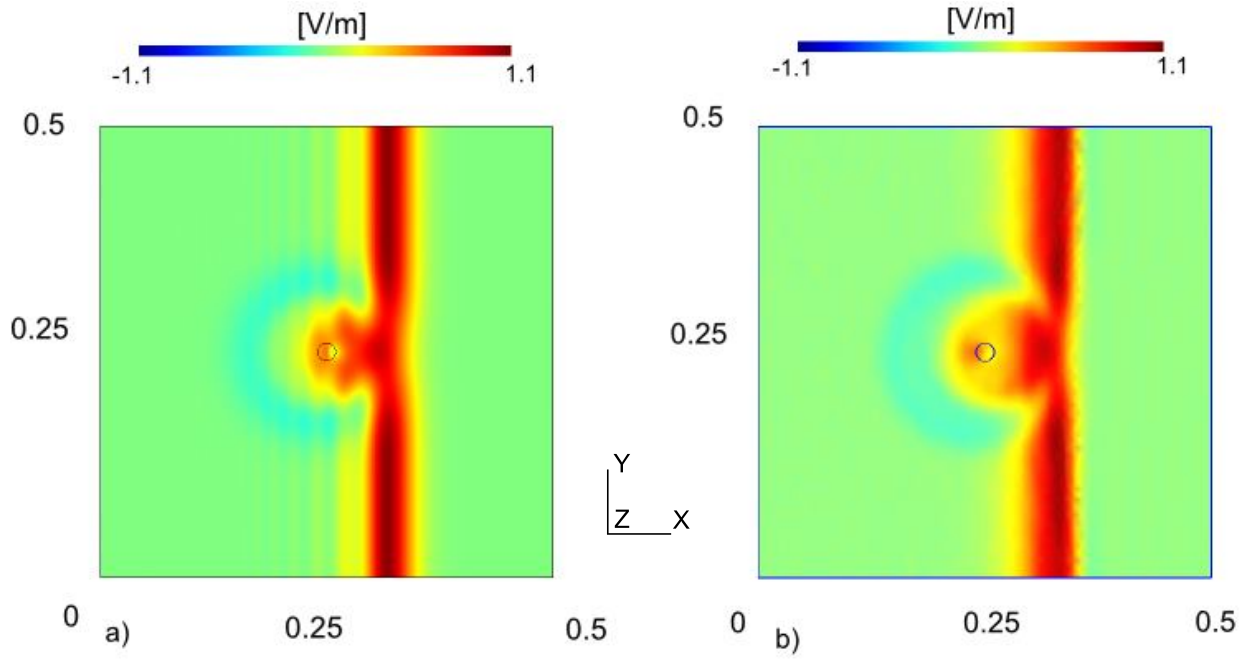
Obr. 29 *Rozložení složky Ez elektrického pole v čase t = 0.74 ns pro dielektrický kruh průměru 0.2 m o permitivitě ε_r = 10 pomocí programů a) COMSOL, b) GETDP*



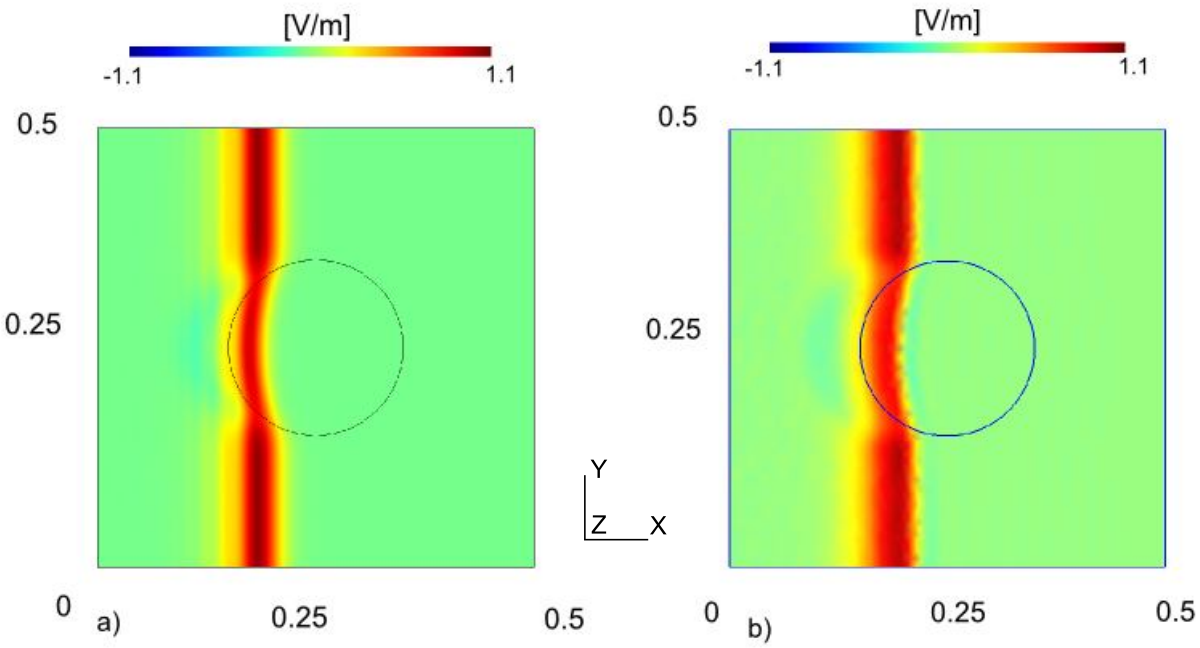
Obr. 30 *Rozložení složky Ez elektrického pole v čase t = 1.2 ns pro dielektrický kruh průměru 0.2 m o permitivitě ε_r = 10 pomocí programů a) COMSOL, b) GETDP*



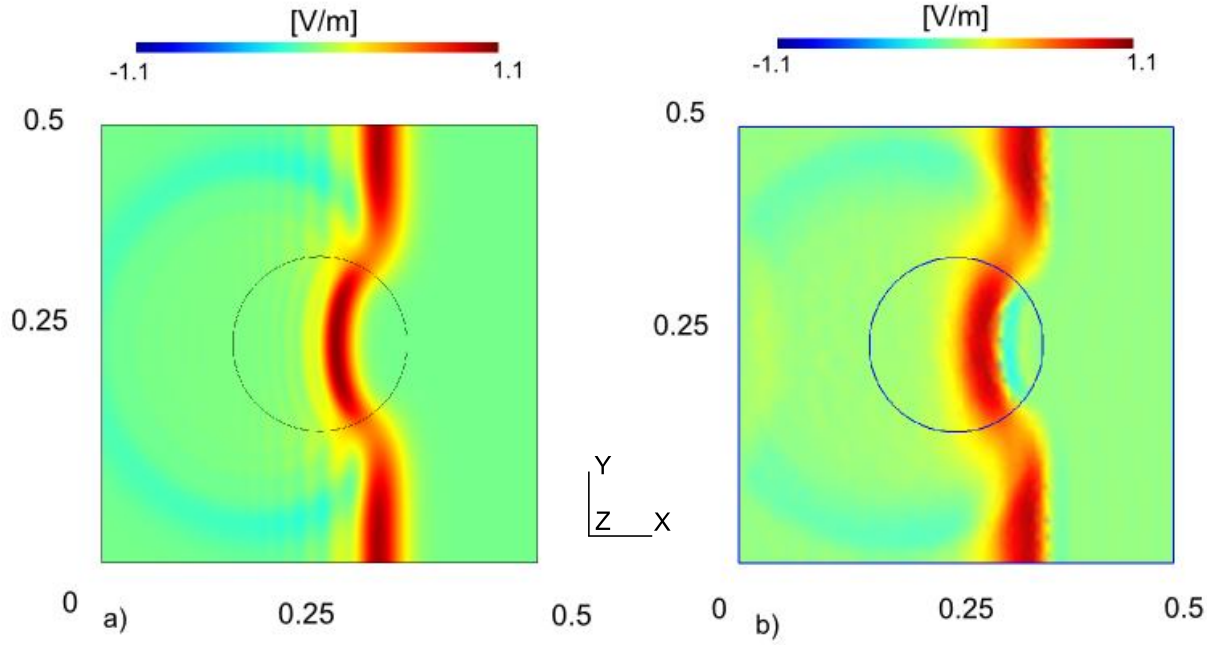
Obr. 31 *Rozložení složky Ez elektrického pole v čase t = 0.92 ns pro dielektrický kruh průměru 0.02 m o permitivitě $\varepsilon_r = 10$ pomocí programů a) COMSOL, b) GETDP*



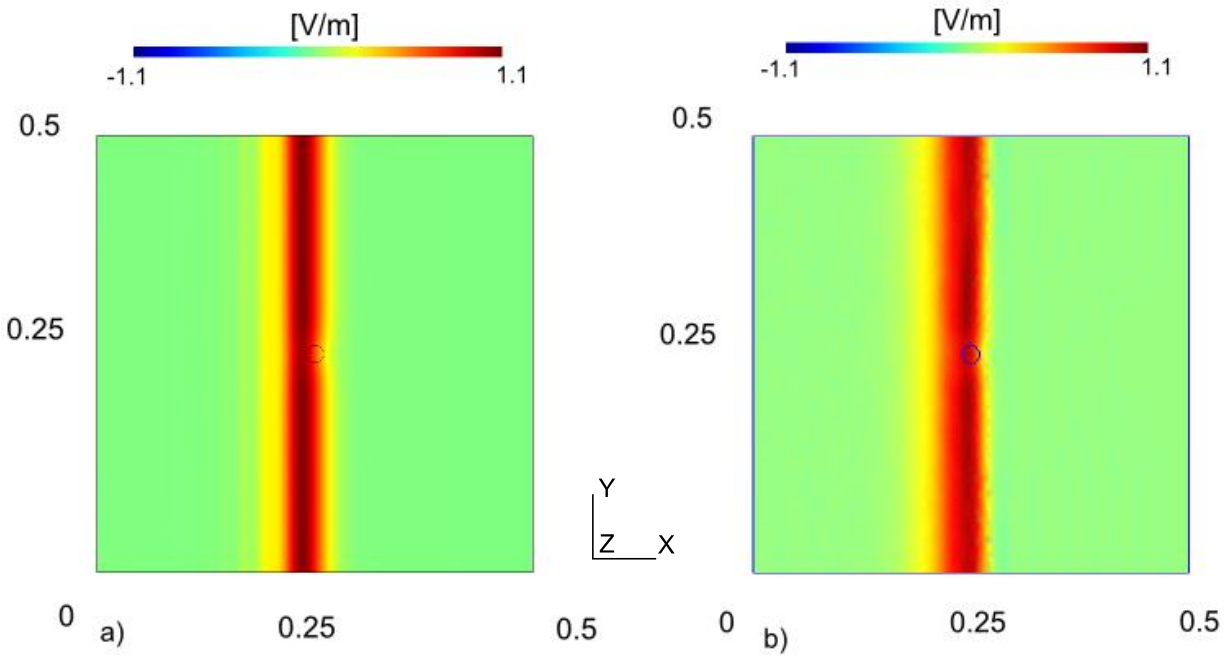
Obr. 32 *Rozložení složky Ez elektrického pole v čase t = 1.2 ns pro dielektrický kruh průměru 0.02 m o permitivitě $\varepsilon_r = 10$ pomocí programů a) COMSOL, b) GETDP*



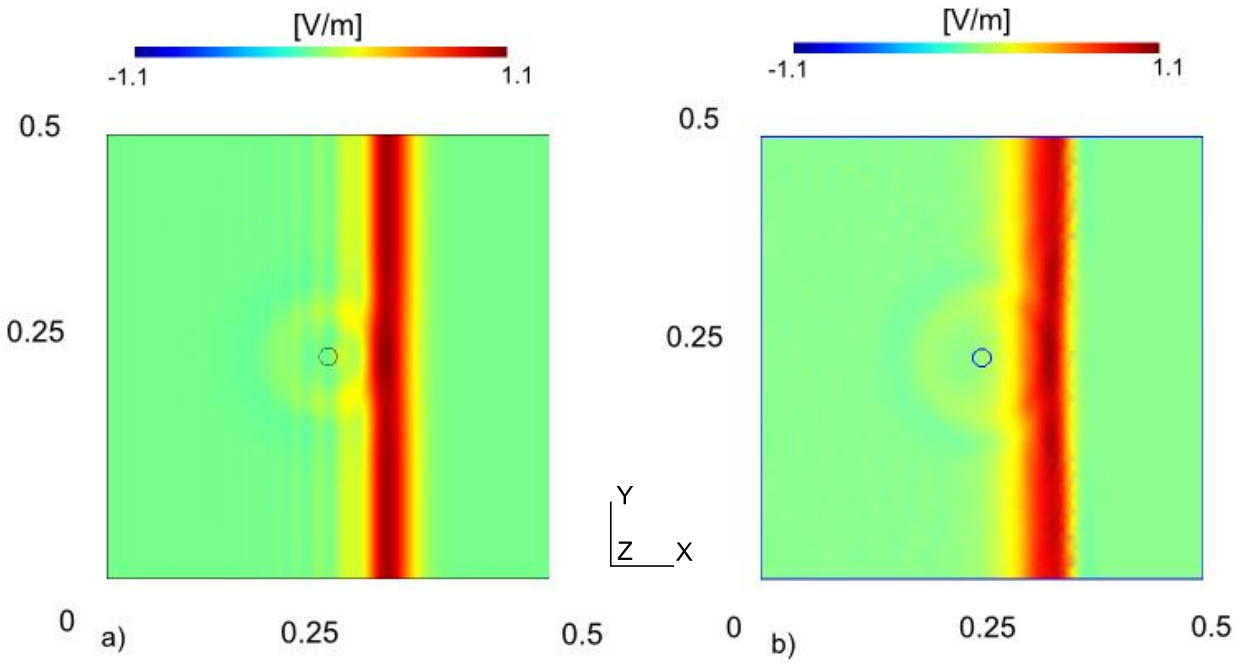
Obr. 33 *Rozložení složky Ez elektrického pole v čase t = 0.74 ns pro dielektrický kruh průměru 0.2 m o permitivitě $\varepsilon_r = 2$ pomocí programů a) COMSOL, b) GETDP*



Obr. 34 *Rozložení složky Ez elektrického pole v čase t = 1.2 ns pro dielektrický kruh průměru 0.2 m o permitivitě $\varepsilon_r = 2$ pomocí programů a) COMSOL, b) GETDP*



Obr. 35 *Rozložení složky Ez elektrického pole v čase t = 0.92 ns pro dielektrický kruh průměru 0.02 m o permitivitě $\varepsilon_r = 2$ pomocí programů a) COMSOL, b) GETDP*



Obr. 36 *Rozložení složky Ez elektrického pole v čase t = 1.2 ns pro dielektrický kruh průměru 0.02 m o permitivitě $\varepsilon_r = 2$ pomocí programů a) COMSOL, b) GETDP*

Na Obr. 31 až Obr. 38 je zachyceno šíření elektromagnetické vlny, kterému jsou do cesty ve směru šíření postupně vkládány různé velikosti dielektrické překážky s rozdílnou relativní permitivitou. Velikost hodnoty permitivity dielektrické překážky má vliv především na rychlost šíření vlny touto překážkou. Na Obr. 31 až Obr. 34 je dielektrická překážka charakterizována hodnotou relativní permitivity $\varepsilon_r = 10$. Je zde možné pozorovat znatelné zpomalení vlny při průchodu překážkou. Toto zpomalení vlny je také závislé na velikosti plochy, kterou překážka vytváří. Pro porovnání je také zobrazena dielektrická překážka s hodnotou relativní permitivity $\varepsilon_r = 2$. Zde je zpomalení šíření elektromagnetické vlny již méně patrné. Obrázky jsou zobrazeny ve stejné barevné stupnici, avšak u výsledků pomocí programu GETDP je možné pozorovat ne zcela přesné vykreslení šířící se elektromagnetické vlny. Příčiny mohou být v dokonalejším vizuálním zpracování výsledků pomocí programu COMSOL, nebo také v použité aproximaci při hledání neznáme veličiny. Program COMSOL využívá aproximaci pomocí kvadratické funkce (Lagrangeův interpolační polynom). V případě programu GETDP je možné použít pouze aproximaci pomocí lineární funkce.

11 Závěr

Cílem diplomové práce bylo seznámit se s volně dostupnými programovými prostředky pro analýzu fyzikálních problémů na bázi metody konečných prvků. Na trhu je v dnešní době ohromné množství komerčních programů, které se specializují na různé oblasti využití. Pro malou společnost nebo vlastní užití je však nákup takového programu finančně velmi náročný. Z tohoto důvodu je vhodné využít volně dostupné programové prostředky.

V průběhu práce byly prostudovány volně dostupné grafické editory pro návrh analyzované struktury, generátory sítí pro vytvoření diskretizační sítě analyzované oblasti. Dále vhodný řešič, který je schopen vyřešit velké množství fyzikálních problémů a také program pro vizuální zobrazení výsledků řešení.

Z těchto dílčích programů byla vybrána nejvhodnější sada nástrojů tvořící kompletní simulační prostředí. Hlavními parametry při výběru vhodného dílčího nástroje bylo zvoleno množství volně přístupné dokumentace, počet nabízených funkcí a také množství formátů pro exportovaná data.

Jako nejvýhodnější se jeví sada nástrojů z obr. 6. Byl vybrán grafický editor FreeCad, který poskytuje pro uživatele příjemné prostředí, vzdáleně připomínající AutoCad. Generátorem sítě byl zvolen program SALOME Platform, případně Gmsh MESH modul. Oba programy umožňují 1D, 2D a 3D algoritmy generování sítě. Jako řešič byl vybrán GETDP, který nabízí dobře zpracovanou dokumentaci a velké množství předem připravených funkcí, urychlujících vytváření kódu. Pro grafické znázornění výsledku simulace byl vybrán modul Gmsh postprocessing, případně program Paraview, který nabízí propracované prostředí, podporující velké množství vstupních formátů dat.

Dále jsem v práci prozkoumal vzájemné propojení různých programů pomocí exportu dat a následným načtením v jiném programu. Grafické editory struktur a generátory sítí mají velké množství formátů pro přenos dat. Čím více se však v simulačním řetězci pohybujeme vzhůru, tím možnost vzájemné výměny dat klesá. Různé programy, mezi které patří i Elmer, TetGen, apod. neumožňují export dat. Od počátku simulačního procesu až po vizualizaci výsledku využívají svoje vlastní specifické formáty.

Poslední kapitola diplomové práce se zabývá ověřením výše vybrané sady volně šiřitelných nástrojů. Byla zde implementována sada ukázkových úloh pro řešení elektromagnetických struktur. Zde jsem vycházel také z vybraných příkladů, které jsou uvedeny v počítačových cvičeních k předmětu Elektromagnetické vlny, vedení a antény [18]. Návrh geometrie řešeného problému a vygenerování diskretizační sítě je velmi intuitivní. Pro definici problému v prostředí řešiče GETDP je však práce již náročnější a vyžaduje použití mnoha funkcí a objektů, které si mezi sebou předávají vypočtená data. Definice problému se vytváří v prostředí textového editoru, které může mnoho uživatelů odradit. Je to však malá cena, vzhledem k ceně licence komerčního software. Řešič GETDP nabízí velké množství možností, pro jejichž použití je však v jistých případech nutná hlubší znalost problematiky řešené úlohy. Ovládání řešiče je umožněno pro zkušenější uživatele z prostředí příkazového řádku, avšak existuje i možnost ovládat dílčí kroky výpočtu z grafického rozhraní.

Vyřešené dílčí úlohy jsou uvedeny v kapitole 10. Jedná se o úlohy z oblasti elektrostatiky (rozložení elektrostatického pole v okolí mikropásku) a šíření elektromagnetických vln (v časové a frekvenční oblasti, deformace vlny na štěrbině a v okolí dielektrické překážky). K porovnání správnosti řešení byl vybrán komerční software COMSOL Multiphysics. Při porovnání odpovídajících výsledků programů GETDP a COMSOL je zřejmé, že implementace řešených úloh je správná.

12 Literatura

[1] Getfem++ [online], Dostupné na WWW:
<http://download.gna.org/getfem/html/homepage/>

[2] Metoda konečných prvků [online], Dostupné na WWW:
<http://wood.mendelu.cz>

[3] NETGEN [online], Dostupné na WWW:
<http://www.hpfem.jku.at/netgen>

[4] VILČINSKÝ,P. Preprocesor pro kombinované modely. Diplomová práce, FM TUL, Liberec, 2008

[5] C. Geuzaine and J.-F. Remacle, *GMSH: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities*. International Journal for Numerical Methods in Engineering, Volume 79, Issue 11, pages 1309-1331, 2009

[6] TOMICOVÁ,J. Přehled volně dostupného programového vybavení pro řešení parciálních diferenciálních rovnic. Bakalářská práce, ČVUT , Praha, 2007

[7] The Salome Open Source CAE Platform [online]. Dostupné na WWW:
<http://conferences.esa.int/05c26/Seminar-2005-09-28-SALOME.pdf>

[8] Ansys [online]. Dostupné na WWW:
<http://www.svsfem.cz>

[9] Cosmos [online]. Dostupné na WWW:
<http://home.zcu.cz/~msvantne/Techskop/2009_mtp_cviceni_prednaska.html>

[10] Salome Platform [online]. Dostupné na WWW:
<http://www.salome-platform.org/>

[11] CAE Linux [online]. Dostupné na WWW:
<http://caelinux.com/>

[12] FEniCS [online]. Dostupné na WWW:
<http://www.fenics.org>

[13] RIBES, A., CAREMILI, C., *Salome platform component model for numerical simuaion*, Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International, Volume 2, 2007, p. 553-564

[14] Open Source [online]. Dostupné na WWW:
<http://www.pooh.cz/aktovka-x/a.asp?a=2006553>

[15] GETDP [online]. Dostupné na WWW:
< http://www.geuz.org/getdp/>

[16] ŠEDĚNKA, V. *Syntéza mikrovlnných rezonátorů.* Brno: Vysoké učení technické v Brně,Fakulta elektrotechniky a komunikačních technologií, 2009. 42 s., 4 s. příloh. Vedoucídiplomové práce prof. Dr. Ing. Zbyněk Raida.

[17] FASORA, P. *Modelování vlnovodů metodou konečných prvků v časové oblasti.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 38 s. Vedoucí diplomové práce prof. Dr. Ing. Zbyněk Raida.

[18] RAIDA, Z. *Elektromagnetické vlny, vedení a antény – počítačové cvičení.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií.

[19] TETGEN [online], Dostupné na WWW:
< http://tetgen.berlios.de>

[20] Arpack [online], Dostupné na WWW:
< http://www.caam.rice.edu/software/ARPACK/>

[21] Multimediální učebnice [online], Dostupné na WWW:
< http://www.urel.feec.vutbr.cz/~raida/multimedia/index.php>

[22] Kurz k předmětu MMEM [online], Dostupné na WWW:
< http://home.zcu.cz/~karban/teaching/mmem/dolezel/course_1/LECT03.pdf>

[23] Elektromagnetické vlny [online], Dostupné na WWW:
< http://fyzika.gymsusice.cz/web/data/texty/Elmagneticke_vlny_2.pdf>

[24] Vlastní čísla a vlastní vektory [online], Dostupné na WWW:
http://www.studopory.vsb.cz/studijnimaterialy/MatematikaI/13_MI_KAP%202_6.pdf

[25] RAIDA, Z. *Elektromagnetické vlny, vedení a antény – počítačové cvičení.* Úloha č. 1 – Vlnová rovnice, rovinná vlna [online], Dostupné na WWW:
< http://www.urel.feec.vutbr.cz/~raida/beva/>

[26] RAIDA, Z. *Elektromagnetické vlny, vedení a antény – počítačové cvičení.* Úloha č. 4 –vlnovody [online], Dostupné na WWW:
< http://www.urel.feec.vutbr.cz/~raida/beva/>

[27] RAIDA, Z. *Elektromagnetické vlny, vedení a antény – počítačové cvičení.* Úloha č. 4 – difrakce [online], Dostupné na WWW:
< http://www.urel.feec.vutbr.cz/~raida/beva/>

13 Seznam použitých zkratek a symbolů

CIM	počítač integrovaný pro výrobu
CAM	počítačová podpora výroby
CAE	počítačová podpora inženýrství
CAD	počítačová podpora návrhu
CAP	počítačová podpora programování
CAPP	počítačová podpora plánování procesu
CAQ	počítačová podpora kvality
FEM	metoda konečných prvků (Finite Element Method)
PEC	perfektně elektrický vodič
PMC	perfektně magnetický vodič
PML	dokonale přizpůsobená vrstva
PDE	parciální diferenciální rovnice
ODE	obyčejná diferenciální rovnice
TE	transverzálně (příčně) elektrický vid
TM	transverzálně (příčně) magnetický vid
E	intenzita elektrického pole
H	intenzita magnetického pole
ε_0	permitivita vakua
μ_0	permeabilita vakua
ε	permitivita prostředí
μ	permeabilita prostředí