TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Efektivní výpočet průniků sítí různých dimenzí s využitím Plückerových souřadnic

Diplomová práce

M13000181

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Viktor Friš**
Vedoucí práce: Mgr. Jan Březina, Ph.D.

Liberec 2015

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Efficient computation of intersections for meshes of different dimmension based on Plücker coordinates

Diploma thesis

M13000181

Study programme: N2612 – Electrical Engineering and Informatics
Study branch: 1802T007 – Information technology

Author: **Bc. Viktor Friš**
Supervisor: Mgr. Jan Březina, Ph.D.

Liberec 2015

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2014/2015**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Viktor Friš**
Osobní číslo:	**M13000181**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Efektivní výpočet průniků sítí různých dimenzí s využitím Plückerových souřadnic**
Zadávající katedra:	**Ústav nových technologií a aplikované informatiky**

Zásady pro vypracování:

1. Optimalizace tříd pro výpočet průniku trojúhelníka a čtyřstěnu pomocí Plückerových souřadnic (algoritmus TTPC).
2. Návrh lineárního algoritmu pro výpočet průniků elementů 1D a 2D sítí s elementy 3D sítě založeného na průchodu do šířky a algoritmu TTPC.
3. Implementace algoritmu pro průnik sítí.
4. Testování implementace, ověření efektivnosti algoritmu.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **cca 40 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] NIKOS, Platis; THEOHARIS, Theoharis. Fast Ray-Tetrahedron Intersection using Plücker Coordinates. Journal of graphics, gpu and game tools. 2003, vol. 8, num. 4, pp. 37-48. Dostupné z: http://users.uop.gr/nplatis/files/PlatisTheoharisRayTetra.pdf

[2] DORST, Leo; FONTIJNE, Daniel; MANN, Stephen. Geometric algebra for computerscience: an object-oriented approach to geometry. San Francisco: Morgan Kaufmann, 2007, xxxv, 626 p. Morgan Kaufmann series in computer graphics and geometric medeling. ISBN 01-237-4942-5

[3] STROUSTRUP, Bjarne. The C programming language. Special ad. Reading, Mass.: Addison-Wesley, c2000, x, 1019 p. ISBN 02-017-0073-5

Vedoucí diplomové práce: **Mgr. Jan Březina, Ph.D.**
Ústav nových technologií a aplikované informatiky

Datum zadání diplomové práce: **20. října 2014**

Termín odevzdání diplomové práce: **15. května 2015**

V. Kopecký

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci - 14 -

L.S.

prof. Dr. Ing. Jiří Maryška, CSc.
vedoucí ústavu

V Liberci dne 20. října 2014

Prohlášení

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

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 11. 5. 2015

Podpis: Viktor Zuiš

Poděkování

Rád bych poděkoval vedoucímu práce Mgr. Janu Březinovi, Ph.D. za pravidelné a obsáhle konzultace při vytváření diplomové práce. Dále bych rád poděkoval své rodině, své přítelkyni a její rodině za podporu a motivaci práci dokončit.

Abstrakt

Pro konečnoprvkové metody byl vyvinut optimalizovaný algoritmus výpočtu průniku úsečky a trojúhelníku s čtyřstěnem, který je postaven na efektivním výpočtu průniku přímky s trojúhelníkem pomocí Plückerových souřadnic. Průniky jsou reprezentovány barycentrickými souřadnicemi na obou elementech. Dále byl vyvinut lineární algoritmus pro výpočet průniků sítí různých dimenzí, který prochází síť sousedních elementů do šířky.

Implementace algoritmu, potřebných tříd a dalších funkcí je provedena v jazyce C++.

Klíčová slova

Výpočetní geometrie, Plückerovy souřadnice, barycentrické souřadnice, metoda konečných prvků

Abstract

For the finite element method was developed optimized algorithm for calculating the intersection of a line and a triangle with a tetrahedron, which is based on the efficient calculating line-triangle intersection using Plücker coordinates. Intersections are represented by barycentric coordinates of the two elements. Furthermore, linear algorithm was developed to calculate the intersections of meshes with different dimensions, which using breadth-first search for meshes with adjacent elements.

The algorithm, the necessary classes and other functions are implemented in C++.

Keywords

Computational geometry, Plücker coordinates, barycentric coordinates, finite element method

Obsah

Seznam obrázků

Seznam tabulek

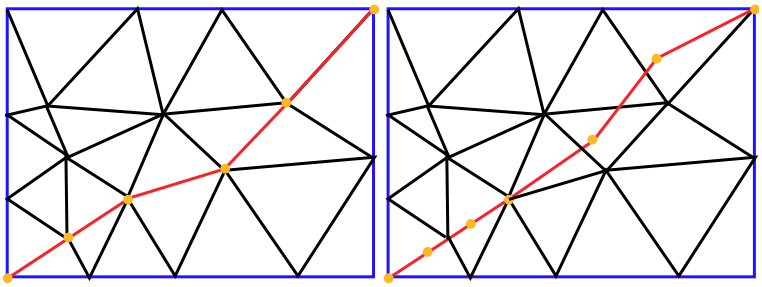
Úvod

Pro metodu konečných prvků jsou potřeba výpočetní sítě, které popisují reálné objekty elementy, nejčastěji popisují objekt čtyřstěny v třírozměrném prostoru. Výpočetní sítě používáme pro úlohy v puklinovém proudění, kde je šířka pukliny mnohem menší než velikost elementu. Reprezentaci puklin a kanálů (1D a 2D nehomogenit) popisujeme pomocí úseček a trojúhelníků v síti čtyřstěnů, čímž nám vznikají sítě s kombinací elementů různých dimenzí.

Program Flow123d, jehož manuál lze nalézt na [1], umí provádět výpočty proudění pro kompatibilní sítě. Kompatibilní sítě jsou popsány čtyřstěny a 1D nebo 2D prvky (kanály nebo pukliny) jsou reprezentovány hranami či stěnami čtyřstěnů. Takovéto sítě je ale komplikované generovat, proto se zabýváme nekompatibilními sítěmi, kde jsou 1D nebo 2D prvky reprezentovány samostatnými úsečkami a trojúhelníky, které prochází skrz čtyřstěny různým způsobem. Příklad velmi malé kompatibilní a nekompatibilní sítě v dvourozměrném prostoru lze vidět na obrázku 1.1. Pro výpočty v nekompatibilních sítích je potřeba výpočet průniků sítí různých dimenzí:

- Úseček s trojúhelníky (dále jen 1D-2D).
- Úseček s čtyřstěny (dále jen 1D-3D).
- Trojúhelníků s trojúhelníky – v práci se tomuto případu nevěnujeme.
- Trojúhelníků s čtyřstěny (dále jen 2D-3D).

V programu Flow123d jsou již implementovány algoritmy pro výpočty průniků 1D-2D, 1D-3D i 2D–3D, které jsou založeny na výpočtu průniků roviny s přímkou pomocí Gaussovy eliminace. Tyto algoritmy ovšem neumí plně reprezentovat průniky a navíc mohou být výpočetně náročné, proto je motivací nového algoritmu plná reprezentace průniků a urychlení výpočtů.



Obrázek 1.1: Příklad kompatibilní a nekompatibilní sítě

Pro výpočty průniků 1D-2D, 1D-3D a 2D–3D jsem se rozhodl použít Plückerovy souřadnice, inspirací pro výpočty mi byl článek [2], který řeší výpočet průniku přímky s trojúhelníkem pomocí Plückerových souřadnic. Tento výpočet rozšiřujeme na výpočet průniku úsečky s trojúhelníkem. Efektivní výpočet průniků pro nekompatibilní sítě tvořené pouze puklinami 1D prvků jsem provedl ve své bakalářské práci [3], ve které se využívá procházení sousedních elementů do šířky a tak se znovu používají data, která už byla jednou vypočtena. V diplomové práci se zaměřuji na výpočet průniků pro nekompatibilní sítě tvořené puklinami 2D prvků v 3D sítí. Algoritmus je založen na nalezení prvního neprázdného průniku 2D pukliny s čtyřstěnem, následuje průchod sítě sousedních elementů do šířky.

Vlastnosti Plückerových souřadnic jsou vysvětleny v kapitole 2, kde jsou vypsány jednotlivé vztahy pro jejich výpočet a další použití. Dále jsou popsány elementy v síti a jejich reprezentace do zobecněné formy trojúhelníku (simplexu) v kapitole 3. V kapitole 4 je popsán výpočet průniku přímky s trojúhelníkem za použití Plückerových souřadnic. Pokud nejdou pro výpočet průniku Plückerovy souřadnice použít, je vysvětlen také výpočet průniku bez jejich použití. Výpočet průniku úsečky s čtyřstěnem je popsán v kapitole 5. Výpočet průniku trojúhelníku s čtyřstěnem je vysvětlen v kapitole 6, kde je navíc probráno trasování výsledného průniku a jeho použití pro další výpočty. Návrhy lineárních algoritmů pro výpočet průniků sítí 1D a 2D elementů uvnitř sítě 3D elementů jsou popsány v kapitole 7. Nejdůležitější vlastnosti datových struktur jsou vypsány v kapitole 8. Optimalizované algoritmy jsou porovnány s podobnými algoritmy z programu Flow123d v kapitole 9, kde jsou i příklady testovaných sítí.

Implementace všech algoritmů, datových struktur a samotný program Flow123d je vytvářen v jazyce C++ [4].

2 Vlastnosti Plückerových souřadnic

Pro efektivní výpočty průniků jsou používány *Plückerovy souřadnice*. Jedná se o jistý šesti-rozměrný vektor souřadnic reprezentující přímku v třírozměrném prostoru. Uvažujme přímku p, určenou bodem A a svým směrovým vektorem U, po té jsou Plückerovy souřadnice přímky p dány vztahem:

$$\pi_p = (U_p, V_p) = (U, U \times A). \tag{2.1}$$

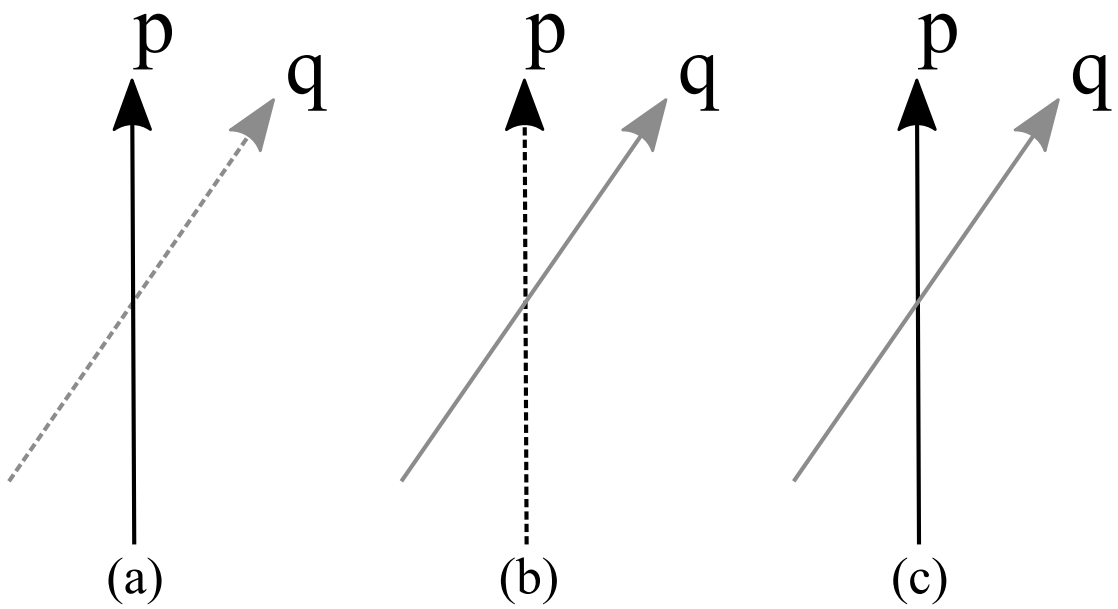
Vzájemnou polohu dvou přímek lze vyjádřit *skalárním součinem* dvou Plückerových souřadnic. Uvažujeme-li přímky p a q, pak skalární součin jejich Plückerových souřadnic je dán vztahem:

$$\pi_p \odot \pi_q = U_p \cdot V_q + U_q \cdot V_p. \tag{2.2}$$

Výsledné znaménko nám určuje orientaci jedné přímky kolem druhé.

- $\pi_p \odot \pi_q > 0$, přímka p obíhá přímku q ve směru hodinových ručiček (viz obrázek 2.1 a).
- $\pi_p \odot \pi_q < 0$, přímka p obíhá přímku q v proti směru hodinových ručiček (viz obrázek 2.1 b).
- $\pi_p \odot \pi_q = 0$, přímka p protíná přímku q nebo je s ní paralelní (viz obrázek 2.1 c).

Z dalších vlastností Plückerových souřadnic, čerpané z knihy [5], lze zjistit, zda-li přímka protíná trojúhelník. Máme-li přímku p a trojúhelník určen přímkami (a,b,c), které jsou orientovány stejným směrem (ve směru nebo proti směru hodinových ručiček vůči středu trojúhelníku) a přímka p protíná trojúhelník, pak mají všechny skalární součiny Plückerovy souřadnice přímky p a Plückerovy souřadnice každé hrany trojúhelníku stejné znaménko. Dokázání některých základních vlastností Plückerových souřadnic lze nalézt v mé bakalářské práci[3].



Obrázek 2.1: Možná relativní orientace dvou přímek p a q

Ze skalárních součinů lze vypočítat *barycentrické souřadnice* průniku na trojúhelníku. Pokud opět uvažujeme přímku p a trojúhelník (v_0, v_1, v_2) a průnik X, pak barycentrické souřadnice průniku jsou určeny vztahem:

$$u_i = \pi_p \odot \pi_{vi} / \sum_{j=0}^{2} \pi_p \odot \pi_{vj}, \tag{2.3}$$

který lze opět nalézt v mé bakalářské práci [3].

Z barycentrických souřadnic průniku na trojúhelníku lze snadno spočíst globální souřadnice průniku na trojúhelníku. Výpočet je dán vztahem:

$$X = \sum_{i=0}^{2} u_i V_i. \tag{2.4}$$

3 Referenční simplex

Elementy ze sítě si převádíme na *simplex*, jež nám popisuje zobecněný trojúhelník. Pro naše účely se zabýváme 0D simplexem (bodem), 1D simplexem (úsečkou), 2D simplexem (trojúhelníkem), 3D simplexem (čtyřstěnem). Každý vytvořený simplex si rekurzivně vytvoří simplexy nižší dimenze až do subdimenze 0, tedy simplex dimenze *D* obsahuje *D+1* simplexů dimenze *D-1*.

Pro práci se simplexem je nutné si zavést vlastní označení vrcholů, hran a stěn. Aby označení bylo konzistentní, zavádíme referenční simplex, kde jsou definována všechna označení pro různé dimenze simplexu. Referenční simplex je vlastně jeden určitý čtyřstěn/trojúhelník/úsečka s definovaným označením a s vrcholy s konkrétními souřadnicemi. Souřadnice vrcholů jsou hodnoty barycentrických souřadnic vzhledem k simplexu (viz tabulka 3.1). V referenčním simplexu si defi-

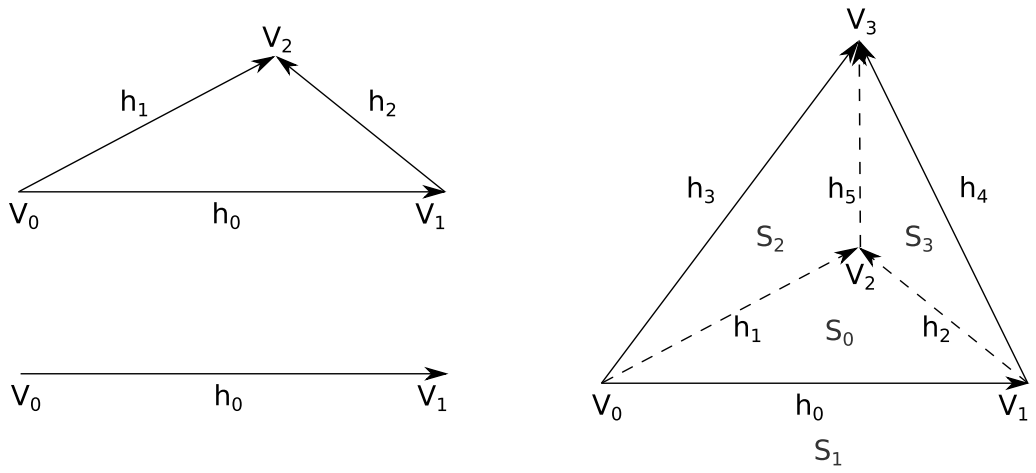
bod	úsečka	trojúhelník	čtyřstěn
$V_0 = (1)$	$V_0 = (1;0)$	$V_0 = (1;0;0)$	$V_0 = (1;0;0;0)$
	$V_1 = (0;1)$	$V_1 = (0;1;0)$	$V_1 = (0;1;0;0)$
		$V_2 = (0;0;1)$	$V_2 = (0;0;1;0)$
			$V_3 = (0;0;0;1)$

Tabulka 3.1: Barycentrické souřadnice referenčních simplexů

nujeme, kolik má simplex vrcholů, hran a stěn. Simplex dimenze *D* obsahuje *D+1* vrcholů, *D+1* stěn a $D(D+1)/2$ hran.

Nejsilnější stránkou referenčního simplexu je ta, že nám sjednocuje orientace hran a stěn pro všechny definované dimenze. Orientace hran plyne právě z označení vrcholů, kde vrchol s nižším indexem je počáteční bod a vrchol s vyšším indexem je koncový bod. Orientace stěn se řídí podle směru normály ke stěně (trojúhelníku). Označení vrcholů, hran a případně stěn je ukázáno na obrázku 3.1 pro úsečku, trojúhelník i čtyřstěn.

Na obrázcích jsou zobrazeny i případné orientace hran. Detailněji jsou orientace



Obrázek 3.1: Označení vrcholů a hrany úsečky v levé dolní části obrázku; označení vrcholů a hran trojúhelníku v levé horní části obrázku; označení vrcholů, hran a stěn čtyřstěnu v pravé části obrázku

a označení vrcholů popsány v tabulkách 3.2, 3.4 a 3.3. Orientace jsou důležité k detekci průniku přímky s trojúhelníkem a proto díky datům z referenčního simplexu můžeme vždy simplexy otáčet a prohazovat jejich subsimplexy tak, aby byly přímky pro trojúhelník vždy ve správné orientaci.

vrcholy úsečky	orientace hrany
V_0,V_1	$h_0(V_0,V_1)$

Tabulka 3.2: Orientace úsečky

vrcholy trojúhelníku	orientace hran	orientace trojúhelníku
V_0,V_1,V_2	$h_0(V_0,V_1)$,$h_1(V_0,V_2)$,$h_2(V_1,V_2)$	směrem ke čtenáři

Tabulka 3.3: Orientace hran v trojúhelníku

Orientace hran v trojúhelníku jsou potom naprosto shodné jako jednotlivé stěny v čtyřstěnu.

stěna	orientace stěny	vrcholy stěny	orientace hran stěny
S_0	dovnitř	V_0,V_1,V_2	$h_0(V_0,V_1)$,$h_1(V_0,V_2)$,$h_2(V_1,V_2)$
S_1	ven	V_0,V_1,V_3	$h_0(V_0,V_1)$,$h_3(V_0,V_3)$,$h_4(V_1,V_3)$
S_2	dovnitř	V_0,V_2,V_3	$h_1(V_0,V_2)$,$h_3(V_0,V_3)$,$h_5(V_2,V_3)$
S_3	ven	V_1,V_2,V_3	$h_2(V_1,V_2)$,$h_4(V_1,V_3)$,$h_5(V_2,V_3)$

Tabulka 3.4: Orientace hran a stěn v čtyřstěnu

4 Průnik přímky s trojúhelníkem

Průnik přímky s trojúhelníkem ve třírozměrném prostoru je založen na efektivním výpočtu průniku přímky s trojúhelníkem pomocí Plückerových souřadnic. Všechny další výpočty průniků vyšších dimenzí (průnik úsečky s čtyřstěnem a průnik trojúhelníku s čtyřstěnem) využívají právě výpočet průniku přímky s trojúhelníkem. Algoritmus nejprve spočte Plückerovy souřadnice pro přímku a hrany trojúhelníku, pokud již nebyly spočteny dříve. Pro přímku s každou hranou trojúhelníku se vypočítá skalární součin jejich Plückerových souřadnic, pokud opět již nebyl spočten dříve. Z definice orientací hran v trojúhelníku (viz 3.1) vyplývá, že hrana s indexem 1 je opačně, kdybychom chtěli trojúhelník orientovat v protisměru hodinových ručiček vůči přímce, proto musíme znaménko skalárního součinu pro hranu (s indexem 1) invertovat. Průnik přímky s trojúhelníkem nastane, když všechny skalární součiny mají stejné znaménko. Pokud by byl některý ze skalárních součinů nulový, nemohli bychom k nalezení průniku použít Plückerovy souřadnice.

4.1 Výpočet průniku s nenulovými skalárními součiny dvou Plückerových souřadnic

Jsou-li spočteny všechny 3 skalární součiny dvou Plückerových souřadnic a všechny mají stejné znaménko a ani jeden není nulový, jedná se o průnik. Průnikem přímky s trojúhelníkem se rozumí jejich *průsečík*. Díky skalárním součinům dvou Plückerových souřadnic můžeme vypočítat k průsečíku další důležité informace, jako jsou:

- Znaménko skalárního součinu dvou Plückerových souřadnic, které dále popisujeme jako *orientaci průsečíku*. Orientace průsečíku udává, jestli je přímka ve směru normály trojúhelníku.
- Barycentrické souřadnice průsečíku na přímce (viz vzorec 4.1).

- Barycentrické souřadnice průsečíku na trojúhelníku (viz vzorec 2.3).

Uvažujme přímku p určenou bodem A a směrovým vektorem U, dále uvažujme bod X, který je průnikem přímky p a trojúhelníku (V_0, V_1, V_2), pak barycentrické souřadnice bodu X na přímce p jsou určeny vztahem:

$$u_0 = 1 + A_i - X_i/U_i, \tag{4.1}$$
$$u_1 = 1 - u_0,$$

kde index i, je index souřadnice směrového vektoru s největší absolutní hodnotou. Tím je zaručeno, že nebudeme dělit nulou a výpočet bude numericky stabilní.

4.2 Výpočet průniku s nulovými skalárními součiny dvou Plückerových souřadnic

Pokud existuje alespoň jeden nulový skalární součin dvou Plückerových souřadnic, nemůžeme k nalezení průniku použít znaménka a hodnoty skalárních součinů. Tyto případy dále v práci označuji jako *speciální případy*. Počet nulových součinů může vyjadřovat následující případy:

- Jeden nulový skalární součin – přímka protíná pouze jednu hranu trojúhelníku.
- Dva nulové skalární součiny – přímka protíná trojúhelník v jeho vrcholu a nikde jinde.
- Tři nulové skalární součiny – obecně všechny ostatní případy. Přímka může protínat všechny tři hrany trojúhelníku, může protínat vrchol trojúhelníku a s třetí hranou být rovnoběžná, přímka může být totožná s hranou trojúhelníku nebo tvořit část hrany trojúhelníku.

Každý nulový skalární součin Plückerových souřadnic přímky a konkrétní hrany trojúhelníku převádíme na hledání společného průsečíku dvou přímek. Uvažujme přímku p, určenou bodem A a směrovým vektorem U, a přímku q, určenou bodem B a směrovým vektorem V, a jejich společný průsečík X. Průsečík X můžeme parametricky vyjádřit výrazy:

$$X = A + sU = B + tV, \tag{4.2}$$

ze kterých po úpravě dostaneme rovnici:

$$sU - tV = B - A. \tag{4.3}$$

K určení parametrů s a t nám slouží 3 rovnice o dvou neznámých. Cramerovým pravidlem lze vypočítat každou z kombinací 2 rovnic o dvou neznámých. Pokud všechny 3 kombinace nemají řešení, přímky nemají společný průsečík. Implementačně je tento způsob optimalizován, aby nedocházelo k výpočtů pro všechny 3 kombinace, hledá se nenulový absolutně maximální determinant ze 3 kombinací a zbytek výpočtu je prováděn pouze pro tuto kombinaci. Pokud je parametr t mimo interval [0,1], průsečík leží mimo trojúhelník a je pro naše účely nezajímavý. Parametr s se převede na barycentrické souřadnice průsečíku na přímce p a parametr t se podle indexu hrany převede na barycentrické souřadnice průsečíku na trojúhelníku.

5 Průnik úsečky s čtyřstěnem

Výpočet průniku úsečky s čtyřstěnem je založen na výpočtu přímky s čtyřstěnem. Inspirací mi byl článek [2]. Algoritmus byl navržen tak, aby počítal průniky nižších dimenzí (průnik přímky s trojúhelníkem s využitím Plückerových souřadnic) a nalezené průsečíky kontroloval, jestli se jedná o průsečíky ležící na úsečce či uvnitř čtyřstěnu. Průnikem mohou být maximálně dva průsečíky.

- **Průnikem je jeden průsečík.** Pokud se jedná pouze o jeden průsečík, tak tento průsečík vznikl speciálním případem (viz 4.2), protože vznikl na hraně nebo ve vrcholu čtyřstěnu. V takovém případě se musí ověřit, jestli jsou barycentrické souřadnice průsečíku na přímce v intervalu [0,1]. Pokud nejsou, nejedná se o průnik úsečky s čtyřstěnem.
- **Průnikem jsou dva průsečíky.** Aby se jednalo o průnik úsečky s čtyřstěnem, mohou nastat 4 různé vzájemné polohy úsečky vůči čtyřstěnu, ze kterých plyne, jestli se opravdu jedná o průnik úsečky s čtyřstěnem. Uvažujme průsečíky P a Q a jejich druhou barycentrickou souřadnici u_p a u_q, po té mohou nastat následující případy vzájemné polohy úsečky vůči čtyřstěnu:
  - $u_p, u_q \in [0, 1]$ – jedná se rovnou o průnik úsečky s čtyřstěnem.
  - $(u_p, u_q > 1 \vee u_p, u_q < 0)$ – celá úsečka leží mimo čtyřstěn, nejedná se o průnik.
  - $((u_p \in [0, 1], u_q \notin [0, 1]) \vee (u_p \notin [0, 1], u_q \in [0, 1]))$ – úsečka protíná čtyřstěn, ale jedním vrcholem začíná nebo končí uvnitř čtyřstěnu. Barycentrické souřadnice takového průsečíku na přímce jsou nastaveny na hodnoty 0/1 a barycentrické souřadnice průsečíku v čtyřstěnu jsou podle barycentrických souřadnic na přímce interpolovány.
  - $((u_p > 1, u_q < 0) \vee (u_p < 0, u_q > 1))$ – celá úsečka leží uvnitř čtyřstěnu, barycentrické souřadnice obou průsečíků na přímce se nastaví na hodnoty 0/1 a obě barycentrické souřadnice průsečíků na čtyřstěnu se interpolují.

Algoritmus si pro každou stěnu čtyřstěnu volá algoritmus pro výpočet přímky s trojúhelníkem, tedy až 4 průchody. Jelikož stačí vypočítat maximálně 2 průsečíky, může se procházení stěn ukončit dříve, pokud již byly oba průsečíky nalezeny. Pokud při třetím průchodu nebyl nalezen ani jeden bod, je procházení čtvrté stěny zbytečné. Pokud se jedná o speciální případy, můžeme vypočíst průsečík pro konkrétní hranu stěny čtyřstěnu a sousední stěnu přes hranu nemusíme již vypočítávat, protože na ni by vznikl stejný průsečík a žádný jiný. Pokud je speciální případ ve vrcholu čtyřstěnu, nemusíme procházet sousední dvě stěny čtyřstěnu.

Jedna ze zásadních optimalizací je předávání vypočtených Plückerových souřadnic a součinů následujícím průchodům stěn. Tudíž pro čtyřstěn se vypočítává maximálně 6 Plückerových souřadnic a 6 skalárních součinů dvou Plückerových souřadnic oproti dvojnásobnému množství.

Každý z průsečíků je jasně definován, na které stěně čtyřstěnu vznikl nebo pokud byl průsečík vrcholem úsečky a byl interpolován. Všechny barycentrické souřadnice průsečíků na stěnách čtyřstěnu se převádí na barycentrické souřadnice průsečíků v čtyřstěnu.

6 Průnik trojúhelníku s čtyřstěnem

Výpočet průniku trojúhelníku s čtyřstěnem vychází z výpočtů průniků přímky a trojúhelníku s využitím Plückerových souřadnic, který je optimalizován předáváním Plückerových souřadnic a skalárních součinů Plückerových souřadnic. Výstupem algoritmu je seznam průsečíků, který tvoří polygon až o 7 vrcholech. Algoritmus byl opět navržen tak, aby počítal průniky nižších dimenzí. Výpočet je rozdělen na dvě části:

- Vypočítají se průniky úsečky s čtyřstěnem pro každou hranu trojúhelníku.
- Vypočítají se průniky přímky s trojúhelníkem pro každou hranu čtyřstěnu.

Průnikem úsečky s čtyřstěnem pro každou hranu trojúhelníku jsou jeden nebo dva průsečíky. Tyto průsečíky jsou navíc definovány hranou trojúhelníku, na které vznikly. Barycentrické souřadnice průsečíku na úsečce jsou převedeny na barycentrické souřadnice průsečíku na trojúhelníku. Pokud je průsečíkem vrchol trojúhelníku uvnitř čtyřstěnu, vypočítá se dvakrát (jednou pro každou úsečku, ke které patří).

Průnikem přímky s trojúhelníkem pro každou hranu čtyřstěnu je vždy maximálně jeden průsečík. Ten je definován hranou čtyřstěnu, na které vznikl. Aby se jednalo o průsečík na čtyřstěnu, musí být jeho barycentrické souřadnice v intervalu [0,1]. Barycentrické souřadnice průsečíku na hraně čtyřstěnu jsou převedeny na barycentrické souřadnice průsečíku v čtyřstěnu. Navíc pro průniky přímky s trojúhelníkem pro každou hranu čtyřstěnu není potřeba vypočítávat průsečíky speciálním případem, protože by se už vypočítaly v první části algoritmu.

Využívá se zde optimalizace předávání už vypočtených Plückerových souřadnic a skalárních součinů. Pokud se vypočítají Plückerovy souřadnice a skalární součiny pro první část algoritmu (průniky hran trojúhelníku s čtyřstěnem), potom pro druhou část jsou všechny znovu použity. Pro celý algoritmus se tedy vypočítá pouze 9 Plückerových souřadnic a 18 skalárních součinů. Kdyby se v hierarchii

výpočtů nepředávaly vypočítaná data, bylo by vypočteno až 63 Plückerových souřadnic a 54 skalárních součinů.

Průnikem trojúhelníku s čtyřstěnem je polygon, který chceme orientovat ve shodě trojúhelníku. Jelikož ale vrcholy polygonu vznikají různě podle orientací hran trojúhelníku a druhá část algoritmu vypočítává vrcholy nezávisle na orientaci trojúhelníku, nebude polygon správně orientován, proto je potřeba polygon *trasovat*. Trasováním máme na mysli uspořádání vrcholů polygonu ve směru trojúhelníku. Některé vrcholy mohou být v seznamu duplicitní, pokud se jedná o vrcholy, které reprezentují vrchol trojúhelníku uvnitř čtyřstěnu, takovéto duplicity je potřeba odstranit.

6.1 Trasování obecného polygonu

Trasování obecného polygonu se dá převést na úlohu nalezení konvexního obalu množiny bodů. Existující algoritmy jsou efektivnější v nalezení konvexního obalu množiny bodů ve 2D, než-li ve 3D. Jelikož vrcholy polygonu můžeme reprezentovat průsečíky s barycentrickými souřadnicemi na trojúhelníku, dostáváme reprezentaci polygonu ve 2D a tím můžeme použít efektivní algoritmus pro hledání konvexního obalu množiny bodů ve 2D. Trasování se provádí až po nalezení všech průsečíků průniku trojúhelníku s čtyřstěnem. Pro nalezení konvexního obalu množiny bodů se používá algoritmus *Monotone chain* [6], který je lehce upraven pro naše účely.

Algoritmus vytvoření konvexního obalu se skládá z několika částí:

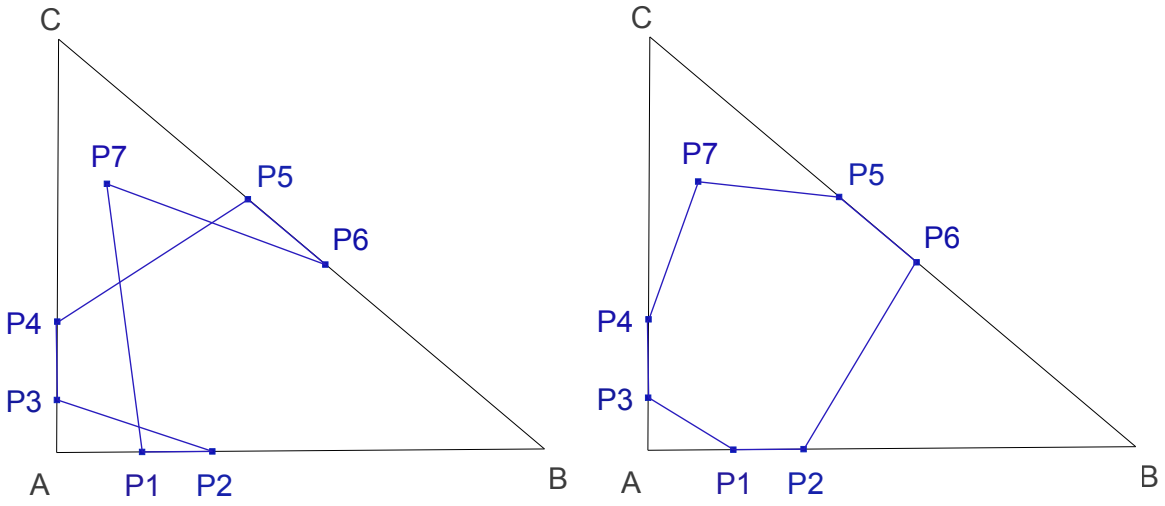
1. **Lexikografické setřídění** - množina průsečíků se lexikograficky setřídí, tzn. průsečíky se setřídí vzestupně podle velikosti 1. barycentrické souřadnice, při shodě 1. barycentrické souřadnice se setřídí vzestupně podle 2. barycentrické souřadnice.

2. **Odstranění duplicit** - průsečíky, které mají stejné obě barycentrické souřadnice, jsou duplicitní a mohou se odstranit. Jelikož je množina průsečíků setříděná, budou duplicity v množině za sebou a jejich nalezení a odstranění lze provést v jednom průchodu.

3. **Vytvoření spodní poloviny konvexního obalu** - definujme si prázdné trasované pole průsečíků H, index i, který označuje právě procházený průsečík, index k, který určuje umístění nově přidávaného průsečíku do trasovaného

pole. Procházejí se všechny setříděné průsečíky od indexu $i = 0$ do n. Ve smyčce se provádí podmínka, pokud je $k >= 2$ a vektorový součin přímek $H_{k-2}H_{k-1} \times H_{k-2}i <= 0$, tedy úhel přímek je konkávní, pak se index k dekrementuje. Po smyčce se na umístění H_k zapíše průsečík i a index k se inkrementuje, tím se vytvoří spodní polovina konvexního obalu.

4. **Vytvoření horní poloviny konvexního obalu** - postup je analogický k vytváření spodní poloviny konvexního obalu. Definujme si navíc index $t = k + 1$. Nyní se prochází setříděné pole průsečíků od indexu $i = n - 2$ do 0. Ve smyčce se provádí podmínka, pokud je $k >= t$ a vektorový součin přímek $H_{k-2}H_{k-1} \times H_{k-2}i <= 0$, pak se index k dekrementuje. Po smyčce se na umístění H_k zapíše průsečík i a index k se inkrementuje. Trasované pole průsečíků tvořené spodní i horní polovinou konvexního obalu se upraví na velikost $k - 1$, tím se docílí správné velikosti pole konvexního obalu.



Obrázek 6.1: Netrasovaný a trasovaný polygon

Na obrázku 6.1 je referenční trojúhelník s vrcholy ABC a s příkladem polygonu, který je tvořen sedmi body P1–P7, před a po trasování.

Trasování polygonu pomocí výpočtu konvexního obalu je vhodné pro všechny obecné polygony, včetně polygonů jejichž průsečíky vznikly speciálním případem, to

jsou takové průsečíky, jejichž součin dvou Plückerových souřadnic vyšel nulový viz 4.2.

6.2 Optimalizované trasování polygonu

Hlavní nevýhody obecného trasování polygonu jsou:

- nutné další výpočty (vektorové součiny),
- nevyužití dodatečných informací z výpočtů průniků,
- složitost $O(NlogN)$.

Proto jsem navrhl čistě kombinatorický algoritmus trasování polygonu, který využívá dodatečné informace z průběhu výpočtů průniků, bez nutnosti dalších výpočtů. Algoritmus je určen pouze pro polygony, které neobsahují průsečíky vzniklé speciálním případem viz 4.2.

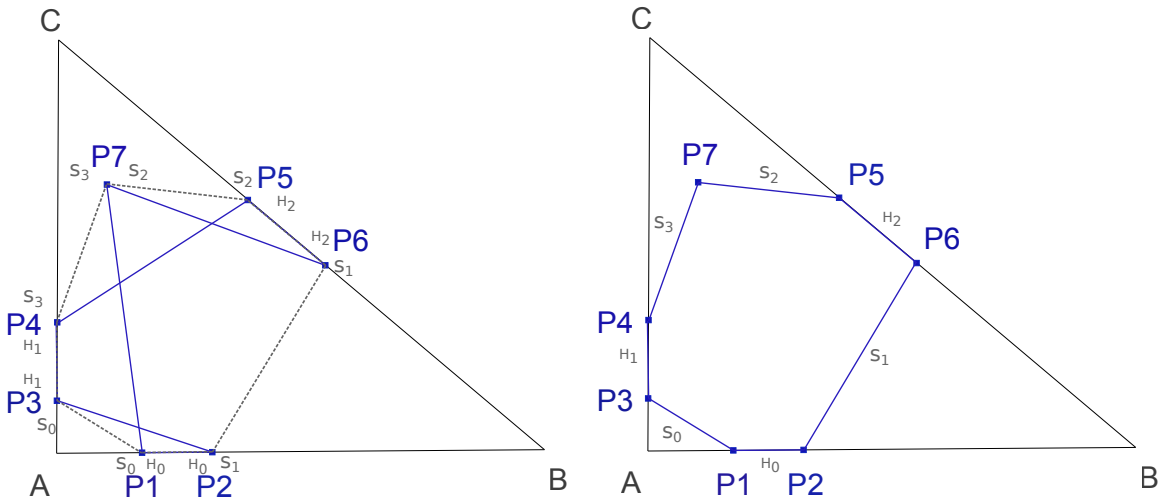
Každý průsečík polygonu vznikl nejdříve průnikem přímky s trojúhelníkem (hrany čtyřstěnu s trojúhelníkem nebo hrany trojúhelníku s každou stěnou čtyřstěnu). Takový průsečík je reprezentován barycentrickými souřadnicemi na obou elementech a *orientací průsečíku* viz 4.1. Pokud je průsečík dále zpracováván průniky vyšších dimenzí, je definován i indexy hran a stěn elementů, na kterých vznikl, proto můžeme rozdělit průsečíky trojúhelníku s čtyřstěnem na tři typy:

- SH - průsečík byl zpracováván na průnik úsečka–čtyřstěn a dále na průnik trojúhelník–čtyřstěn. Jeho barycentrické souřadnice na obou elementech jsou v intervalu (0,1), postupně mu byl přidělen index hrany trojúhelníku a stěny čtyřstěnu, na které vznikl. Podle *orientace průsečíku*, normál stěn v čtyřstěnu a uspořádání hran v trojúhelníku lze určit, jestli průsečík začíná na hraně trojúhelníku či stěně čtyřstěnu, tzn. jestli se jedná o průsečík S–H nebo H–S.
- SS - průsečík byl zpracováván rovnou na průnik trojúhelník–čtyřstěn, tudíž vznikl průnikem hran čtyřstěnu s trojúhelníkem. Průsečíku byl přidělen jen index hrany čtyřstěnu, pomocí kterého lze určit, jaké dvě stěny čtyřstěnu hrana spojuje, podle referenčního čtyřstěnu. Pořadí dvou stěn závisí na *orientaci průsečíku*.

- HH - průsečík byl zpracováván na průnik úsečka–čtyřstěn, kde barycentrické souřadnice průsečíku na úsečce byly rovné 0 nebo 1. Dalším zpracováním na průnik trojúhelník–čtyřstěn průsečík reprezentuje vrchol trojúhelníku uvnitř čtyřstěnu. Jaké dvě hrany trojúhelníku vrchol spojuje lze zjistit z převodní tabulky z referenčního trojúhelníku. Pořadí hran už ale nezávisí na *orientaci průsečíku*, ale je vždy pevně určeno ve směru trojúhelníku.

Samotné trasování spočívá v uspořádání průsečíků tak, aby na sebe navazovaly stejnou hranou trojúhelníku nebo stejnou stěnou čtyřstěnu. Každý průsečík je tedy definován trojicí indexů:

1. index označuje vstupní hranu nebo stěnu
2. index průsečíku v netrasovaném polygonu
3. index označuje výstupní hranu nebo stěnu



Obrázek 6.2: Netrasovaný a optimalizovaně trasovaný polygon

Na obrázku 6.2 lze vidět netrasovaný polygon s průsečíky, které mají u sebe napsané indexy stěn a hran, ke kterým náleží. Pro tento případ by trojice indexů byla definována následovně v tabulce 6.1.

1. index	index průsečíku v netrasovaném polygonu	3. index
S_0	P1	H_0
H_0	P2	S_1
H_1	P3	S_0
S_3	P4	H_1
H_2	P5	S_2
S_1	P6	H_2
S_2	P7	S_3

Tabulka 6.1: Příklad trojic indexů pro průsečíky

Pro optimalizované spojování průsečíků zavádím pomocnou trasovací tabulku. Trasovací tabulka 7×2 reprezentuje svými řádky stěny čtyřstěnu (S_0, S_1, S_2, S_3) a po té hrany trojúhelníku (H_0, H_1, H_2). První sloupeček obsahuje výstupní hranu nebo stěnu, ke které jé vázán průsečík a druhý sloupeček obsahuje index průsečíku v netrasovaném polygonu.

index řádku	index následujícího řádku	index průsečíku
0 (S_0)	4 (H_0)	P1
1 (S_1)	6 (H_2)	P6
2 (S_2)	3 (S_3)	P7
3 (S_3)	5 (H_1)	P4
4 (H_0)	1 (S_1)	P2
5 (H_1)	0 (S_0)	P3
6 (H_2)	2 (S_2)	P5

Tabulka 6.2: Příklad trasovací tabulky

Každý průsečík se podle svého prvního indexu, definujícího vstupní hranu nebo stěnu, zapíše na příslušný řádek v trasovací tabulce, zapíše na něj i svůj index průsečíku a index výstupní hrany nebo stěny. Tímto způsobem se rovnou odstraní duplicitní průsečíky, protože se zapíšou na stejné místo a tím se přepíší a použijí pouze jednou. Trasovaný polygon se sestaví procházením trasovací tabulky, kdy se začne na prvním neprázdném řádků a pokračuje se na další řádek podle výstupního indexu, dokud se nenarazí na řádek, na kterém se začalo. Sestavená trasovací tabulka pro netrasovaný polygon na obrázku 6.2 lze vidět v tabulce 6.2.

Z netrasovaného polygonu P1–P2–P3–P4–P5–P6–P7 se sestaví správně trasovaný polygon P1–P2–P6–P5–P7–P4–P3, který lze opět vidět na obrázku 6.2. Nejen, že trasování není závislé na pomocných výpočtech, ale i každá hrana polygonu je jasně definovaná hranou trojúhelníku nebo stěnou čtyřstěnu, ke které náleží.

7 Lineární algoritmus pro výpočet průniků sítí různých dimenzí

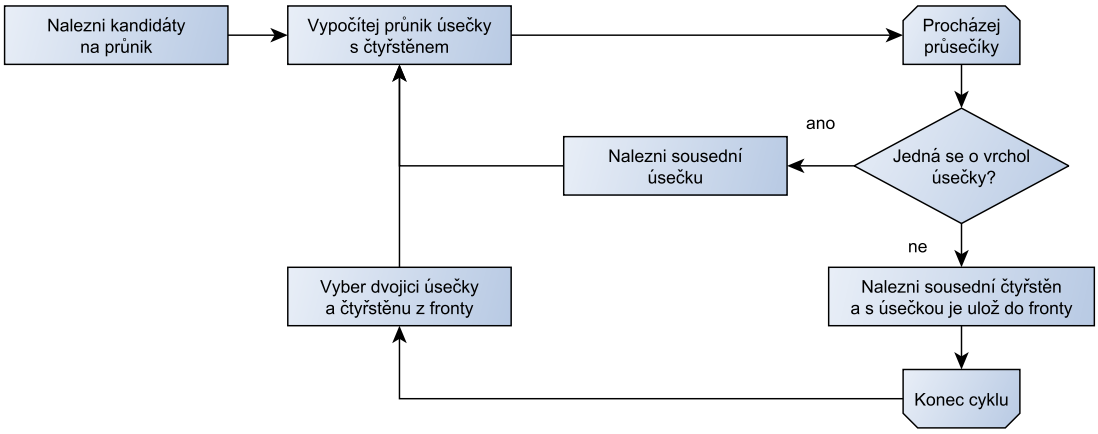
Sítě různých dimenzí máme na mysli sítě úseček a trojúhelníků v síti čtyřstěnů. Více nezávislých sítí úseček a čtyřstěnů označujeme jako komponenty. Průnikem sítí různých dimenzí se rozumí výpočet průniků elementů z komponent s čtyřstěny.

Algoritmy pro výpočet průniků sítí jsou založeny na nalezení prvního neprázdného průniku. Pokud předpokládáme, že všechny komponenty budou uvnitř sítě čtyřstěnů, bude k nalezení prvního neprázdného průniku pro každou komponentu zapotřebí tolik průchodů přes všechny čtyřstěny, kolik je komponent. Každý průchod je realizován pomocí hledání průniku obalových boxů prvků z komponent s čtyřstěny, pokud obalové boxy interagují, vypočítá se pro dvojici elementů průnik. Pokud je průnik neprázdný, pokračování v průchodu je už zbytečné. Čím více bude v síti jednotlivých komponent, které nebudou mít mezi sebou žádný společný element ani žádný společný vrchol elementu, tím pomalejší tato část bude, předpokládáme ale, že síť bude mít až o 4 řády více čtyřstěnů než komponent. Pro atypické sítě, které by měly řádově více komponent, ať už uvnitř sítě čtyřstěnů nebo i mimo, lze k nalezení prvního neprázdného průniku využít algoritmy z programu Flow123d, které jsou založeny na metodě *Bounding interval hierarchy* (viz článek [8]), tyto algoritmy dále v práci uvádím pod zkratkou *BIH Tree*. Algoritmy BIH Tree jsou ovšem pomalé pro běžné sítě s menším počtem komponent, vyplatí se pouze pro případy, kdy jsou elementy z komponent mimo síť čtyřstěnů, ale zároveň jsou uvnitř obalového boxu celé sítě čtyřstěnů a je jich řádově více.

Za účelem rychlého výpočtu, zda-li jsou elementy z komponent či celá komponenta mimo síť čtyřstěnů, vytváří se obalový box celé sítě čtyřstěnů, pokud pak obalový box elementu neinteraguje s obalovým boxem celé sítě, element leží mimo síť čtyřstěnů.

7.1 Výpočet průniků sítí 1D–3D

Výpočtu průniku sítí 1D–3D jsem se věnoval ve své bakalářské práci [3]. Algoritmus bylo potřeba sjednotit a přeimplementovat do nových optimalizovaných struktur, čímž došlo ke zjednodušení celého algoritmu.



Obrázek 7.1: Vývojový diagram algoritmu výpočtu průniků pro sítě 1D–3D

Po nalezení kandidátů úsečky a čtyřstěnu se vypočítá jejich průnik pomocí výpočtů průniků nižších dimenzí (viz kapitola 5). Pokud je průnik neprázdný, je tvořen až dvěma průsečíky, které se dále procházejí a zpracovávají. Průsečíky mohou být dvou typů:

- Průsečík je koncovým bodem úsečky a je uvnitř čtyřstěnu. Barycentrická souřadnice průsečíku je 0 nebo 1 a tvoří počáteční/koncový bod úsečky, pomocí kterého se najdou všechny sousední úsečky a pro ně se rekurzivně počítá průnik s aktuálním čtyřstěnem a proces zpracování průniku se opakuje. Aby nedošlo k zacyklení rekurzí, je zapotřebí si určit, ke které úsečce byl nalezen alespoň jeden neprázdný průnik a při hledání sousedních úseček kontrolovat, jestli je úsečka bez průniku.

- Průsečík vznikl na stěně čtyřstěnu. Barycentrické souřadnice průsečíku jsou v intervalu (0,1). Pomocí indexu stěny čtyřstěnu se nalezne sousední čtyřstěn a výpočet průniku pro konkrétní úsečku a sousední čtyřstěn se uloží do fronty k pozdějšímu zpracování. Aby se nestalo, že se bude opakovat výpočet průniku pro stejnou dvojici elementů, zjišťuje se u sousedního elementu, jestli už netvoří průnik s konkrétní úsečkou. Pokud se už nezpracovává žádný výpočet

průniku rekurzivně, zpracovává se fronta dokud není prázdná. Když se fronta vyprázdní, všechny průniky celé jedné komponenty úseček jsou vypočteny.

Vývojový diagram výpočtu průniků pro sítě 1D–3D je popsán na obrázku 7.1.

7.2 Výpočet průniků sítí 2D–3D

Po nalezení kandidátů trojúhelníku a čtyřstěnu se vypočítá jejich průnik pomocí výpočtů průniků nižších dimenzí (viz kapitola 6). Neprázdný průnik tvoří polygon až o 7 průsečících. Skrze hrany polygonu se prodlužuje do sousedních elementů. K prodloužení je potřeba vytvořit si prodlužovací tabulku, která obsahuje pouze index hrany/stěny elementu a typ elementu (trojúhelník/čtyřstěn). Prodlužovací tabulka může být vytvořena dvěma způsoby:

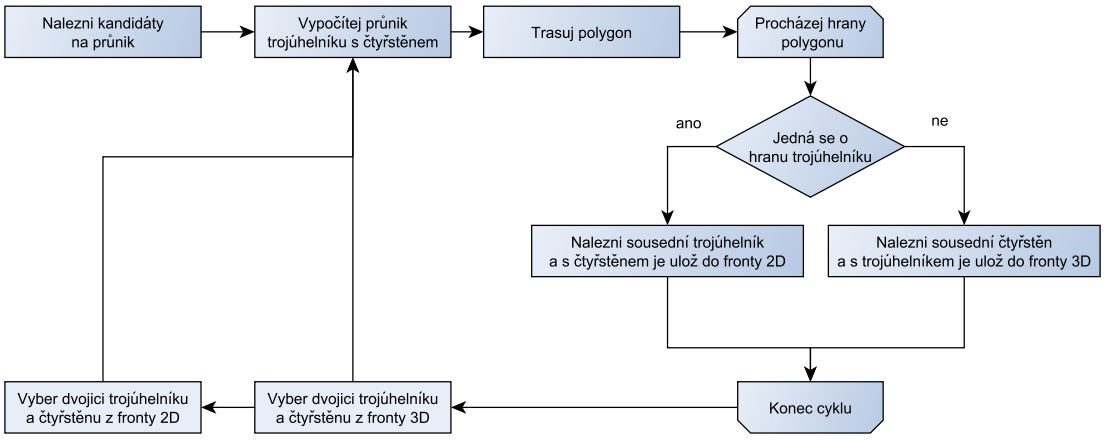
- Polygon obsahuje průsečíky vypočteny speciálním případem (viz podkapitola 4.2). Polygon se bude trasovat pomocí obecného trasování polygonu (viz podkapitola 6.1). Procházením dvojic po sobě jdoucích průsečíků se zjišťuje jaké nulové barycentrické souřadnice na obou elementech mají společné, tím se zjistí na jaké protilehlé hraně/stěně průsečíky leží a tím se vytvoří prodlužovací tabulka. Polygony obsahující průsečíky vypočteny speciálním případem mohou být polygony o jednom nebo dvou průsečících, takové jsou ale pro nás nezajímavé a dále je neprodlužujeme. Pokud se jedná o polygon vytvořený celý na stěně čtyřstěnu, na sousedním čtyřstěnu by byl naprosto stejný a proto ani v takovém případě nehodláme prodlužovat.
- Polygon neobsahuje ani jeden průsečík vypočtený speciálním případem. Polygon se bude trasovat pomocí optimalizovaného trasování polygonu (viz podkapitola 6.2), díky kterému se hned vytvoří i prodlužovací tabulka a není potřeba žádných výpočtů. Takovéto polygony obsahují vždy tři až sedm průsečíků.

Zpracováním prodlužovací tabulky se zjistí, do jakého sousedního elementu se má s výpočtem průniku pokračovat. Vytváříme si dvě fronty (*fronta 2D* a *fronta 3D*), které označují o jaký typ prodloužení se jedná, jestli se prodlužovalo do sousedního trojúhelníku nebo do sousedního čtyřstěnu a podle toho se i průniky dále zpracovávají. Dále se zavádí i příznaky k trojúhelníkům, jestli pro ně byly všechny průniky už spočteny a příznaky pro čtyřstěny, které označují, s kterým trojúhelníkem

naposledy při výpočtu interagovaly, případně mají implicitně nastavenou hodnotu -1.

- Prodlužuje se hranou trojúhelníku. Pomocí indexu hrany trojúhelníku se zjistí jeho sousední trojúhelník. Pokud pro sousední trojúhelník ještě nebyly spočteny všechny průniky nebo trojúhelník nemá s aktuálním čtyřstěnem vypočten průnik, uloží se výpočet průniku do fronty 2D k dalšímu zpracování.
- Prodlužuje se stěnou čtyřstěnu. Pomocí indexu stěny čtyřstěnu se zjistí jeho sousední čtyřstěn. Pokud čtyřstěn ještě neinteragoval s žádným trojúhelníkem (má příznak roven -1) nebo interagoval s jiným trojúhelníkem, než s aktuálním, uloží se výpočet průniku do fronty 3D k dalšímu zpracování.

Nejdříve se zpracovávají všechny průniky z fronty 3D, čímž je dosaženo, že pro konkrétní trojúhelník se naleznou všechny čtyřstěny, které s ním interagují a po té lze trojúhelníku nastavit příznak, že už má všechny průniky vypočteny. Pokud je fronta 3D prázdná, vybere se jeden průnik z fronty 2D a celý proces se opakuje, dokud nejsou obě fronty prázdné. Proces zpracování průniků trojúhelníků s čtyřstěny lze vidět na obrázku 7.2.



Obrázek 7.2: Vývojový diagram algoritmu výpočtu průniků pro sítě 2D–3D

8 Implementace algoritmů a tříd

Veškeré třídy a algoritmy jsou psány v jazyce C++ a implementovány do programu Flow123d. Pro práci s lineární algebrou je používána knihovna *Armadillo*. Ve výpočtech se pracuje s jistou přesností pro double, která se pohybuje mezi 10^{-7} a 10^{-10}. Vytvořené třídy a jejich nejdůležitější vlastnosti jsou:

- **Plucker** - obsahuje šesti-rozměrný vektor Plückerových souřadnic a metody pro jejich výpočet. Rovněž obsahuje metodu pro výpočet skalárního součinu dvou Plückerových souřadnic. Důležitou metodou je kontrola, jestli byly souřadnice v objektu již vypočítány.

- **Simplex** - šablona třídy s parametrem dimenze simplexu. Obsahuje *N+1* subsimplexů, které si vytváří z dodaných bodů. Obsahuje metody na vrácení souřadnic a subsimplexů. Každý simplex s dimenzí větší než nula dokáže vrátit simplex dimenze jedna podle indexu hrany z referenčního simplexu. Simplex dimenze nula obsahuje souřadnice bodu v třírozměrném prostoru.

- **RefSimplex** - šablona třídy s parametrem dimenze simplexu obsahující převážně statická data a statické metody. Obsahuje obecná data o simplexech. Kolik má každá dimenze simplexu hran, vrcholů, stěn, hran na stěnu, vrcholů na stěnu, indexy hran, indexy vrcholů, indexy stěn, orientace hran, orientace stěn atd. Rovněž obsahuje metody na interpolaci dvou barycentrických souřadnic. Obsahuje i metodu na vrácení barycentrické souřadnice vrcholu simplexu různé dimenze.

- **IntersectionPoint** - šablona třídy s parametry dvou dimenzí simplexů. Třída sloužící na uchovávání a manipulaci s daty. Popisuje kompletně průsečík dvou simplexů různé dimenze barycentrickými souřadnicemi na obou elementech, indexy hran a stěn a *orientací průsečíku*, představuje-li průsečík vrchol jednoho ze simplexů nebo jestli průsečík byl nebo nebyl vypočten speciálním případem (viz 4.2).

- **IntersectionLine** - třída reprezentující průnik úsečky s čtyřstěnem. Obsahuje pole průsečíků (IntersectionPoint<1,3>) a indexy úsečky a čtyřstěnu.

- **IntersectionPolygon** - třída reprezentující průnik trojúhelníku s čtyřstěnem. Obsahuje pole průsečíků (IntersectionPoint<2,3>), indexy trojúhelníku a čtyřstěnu a příznak o tom, jestli je alespoň jeden průsečík vypočítán speciálním případem (viz 4.2). Nejdůležitější metoda je trasování polygonu jak obecným, tak optimalizovaným způsobem (viz 6.1 a 6.2). Trasovací metody vytváří prodlužovací tabulky k dalšímu zpracování. Navíc je ve třídě implementována metoda pro výpočet obsahu polygonu, která vrátí obsah polygonu na referenčním trojúhelníku. Výpočet obsahu je proveden rozdělením polygonu na menší trojúhelníky a součtem jejich obsahů.

- **ProlongationPoint** - představuje místo dalšího prodlužování pro průniky úsečky s čtyřstěnem. Uchovává indexy elementů dalšího průniku ke zpracování.

- **ProlongationLine** - představuje místo dalšího prodlužování pro průniky trojúhelníku s čtyřstěnem. Uchovává indexy elementů dalšího průniku ke zpracování a jeho index ve frontě.

- **ComputeIntersection** - šablona třídy s parametry dvou simplexů. Hlavní výpočetní třída. Každá generická třída má inicializační část, ve které si vytváří výpočetní třídy nižších dimenzí a předává jím odkazy na své Plückerovy souřadnice a součiny dvou Plückerových souřadnic. Generické třídy obsahují i specifické výpočetní části, které zpracovávají výsledky z výpočetních částí generických tříd nižších dimenzí. Třída ComputeIntersection<Simplex<1>,Simplex<2>> umožňuje výpočet průniku přímky s trojúhelníkem (viz 4). Třída ComputeIntersection<Simplex<1>,Simplex<3>> umožňuje výpočet průniku úsečky s čtyřstěnem (viz 5). Třída ComputeIntersection<Simplex<2>,Simplex<3>> umožňuje výpočet průniku trojúhelníku s čtyřstěnem (viz 6).

- **InspectElements** - uživatelská třída. Obsahuje algoritmy pro výpočet sítí různých dimenzí (viz 7.1 a 7.2), inicializaci dat (vytváření obalových boxů elementů, příznaků a alokace polí průniků) a práci se sítí (převádění elementů na simplexy). Obstarává prodlužování průniků a dokáže pro průniky trojúhelníků s čtyřstěny vypočítat celkovou plochu všech průniků. Navíc dokáže zapsat původní síť se všemi průniky do souboru a tím síť dále vizualizovat v programu GMSH. Výsledné průniky (IntersectionLine nebo Intersection-

Polygon) se ukládají do pole k indexům úseček nebo trojúhelníků, ke kterým náleží.

Implementační příklad výpočtu průniku jedné dvojice trojúhelníku s čtyřstěnem.

```
Simplex<2> trojuhelnik;
Simplex<3> ctyrsten;
IntersectionPolygon polygon;
ComputeIntersection<Simplex<2>,Simplex<3>> ci(trojuhelnik, ctyrsten);
ci.init();
ci.compute(polygon);
```

V příkladu se neřeší naplnění trojúhelníku a čtyřstěnu daty, uvažujeme, že objekty *trojuhelnik* a *ctyrsten* již mají data.

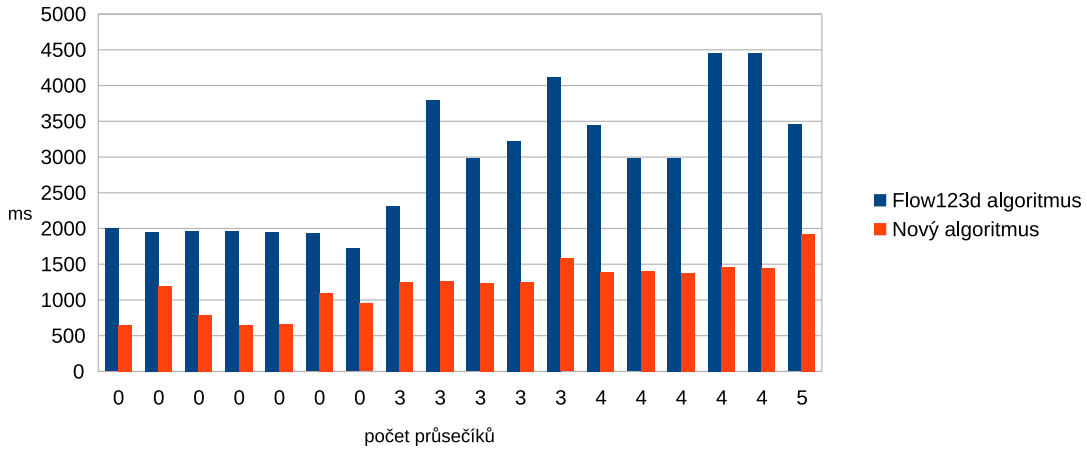
Implementační příklad výpočtu průniků sítí 2D–3D.

```
Mesh sit;
InspectElements ie(&sit);
ie.compute_intersections<2,3>();
ie.print_mesh_to_file("sit");
```

V příkladu se neřeší načtení sítě, uvažujeme, že objekt *sit* již celou síť obsahuje se všemi daty.

9 Výsledky práce

Úspěšně se povedlo implementovat a optimalizovat algoritmus pro výpočet průniku trojúhelníku s čtyřstěnem. Původní algoritmus v programu Flow123d potřeboval pro výpočet zhruba 1296 součinových operací. Algoritmus volal 18× výpočet přímky s trojúhelníkem, kde každý takovýto výpočet prováděl 6 vektorových součinů (1 vektorový součin = 6 součinových operací), 4 skalární součiny (1 skalární součin = 4 součinové operace) a řešil soustavu 3 rovnic o 3 neznámých Gaussovou eliminací pomocí Cramerova pravidla (= 24 součinových operací).



Obrázek 9.1: Graf časové náročnosti výpočtu průniku pro různé dvojice trojúhelníku a čtyřstěnu

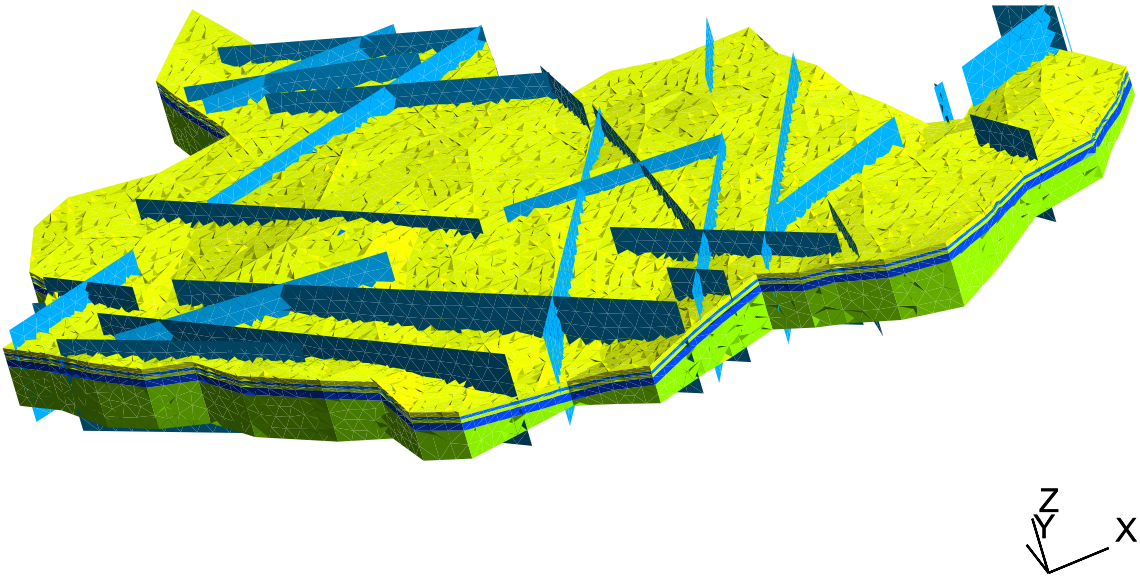
Nový algoritmus provádí odhadem 225 součinových operací. Algoritmus provádí pro výpočet přímky s trojúhelníkem odhadem 45 součinových operací. Pro výpočet trojúhelníku s čtyřstěnem se využívají Plückerovy souřadnice a součin dvou Plückerových souřadnic, které se jednou vypočítají v průběhu procesu a poté se znovu používají. Algoritmus vypočítá maximálně 9 Plückerových souřadnic (1 Plückerova souřadnice = 6 součinových operací) a 18 součinů dvou Plückerových

souřadnic (1 součin dvou Plückerových souřadnic = 6 součinových operací). Průsečíkům se vypočítávají barycentrické souřadnice na úsečce (= 9 součinových operací), takových průsečíků může být maximálně 7.

Na obrázku 9.1 lze vidět časové srovnání výpočtu 18 dvojic trojúhelníku a čtyřstěnu. Dvojice byly náhodně vygenerovány a prvních 7 z nich nemá žádný společný průnik. Kvůli měřitelnosti byl každý výpočet proveden 100000× a naměřené hodnoty byly zaneseny do grafu. Jelikož algoritmus z programu Flow123d při výpočtu průniku rovnou polygon trasuje a vypočítává obsah polygonu, byly pro nový algoritmus při měření nastaveny stejné podmínky. Z grafu vyplývá, že nový algoritmus je zhruba o 59 % rychlejší pro dvojice s neprázdným průnikem a o 55 % rychlejší pro dvojice s žádným společným průnikem.

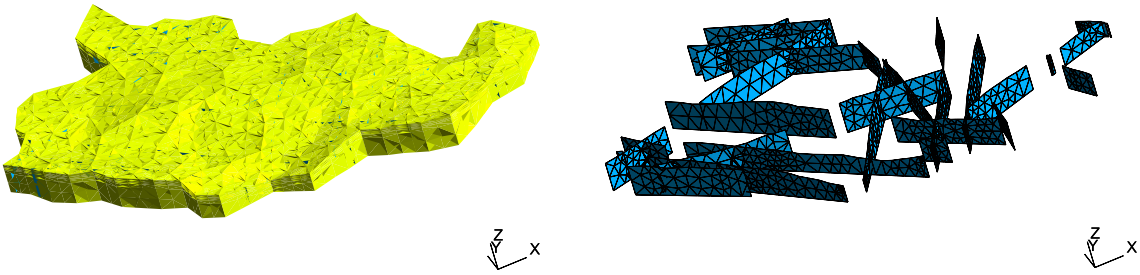
První z testů měření efektivnosti lineárního algoritmu průniků sítí 2D-3D byla provedena pro 5 sítí s různou topologií komponent a různým počtem elementů, z nichž 2 představovaly reálné sítě a 3 byly vygenerovány pro testovací účely.

1. reálná atypická síť na obrázku 9.2, která obsahuje 224371 elementů, 61 komponent tvořeny 5861 trojúhelníky, kde mnoho trojúhelníků je mimo celou síť čtyřstěnů.



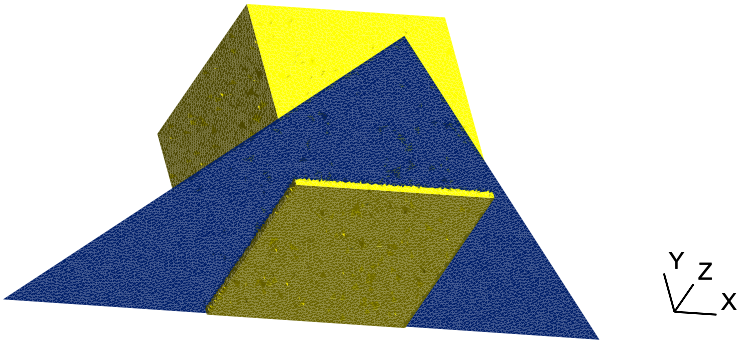
Obrázek 9.2: Příklad 1. testované sítě

2. reálná síť na obrázku 9.3, která obsahuje komponenty pouze uvnitř sítě čtyřstěnů, obsahuje 82843 elementů, 61 komponent tvořeny 1773 trojúhelníky.



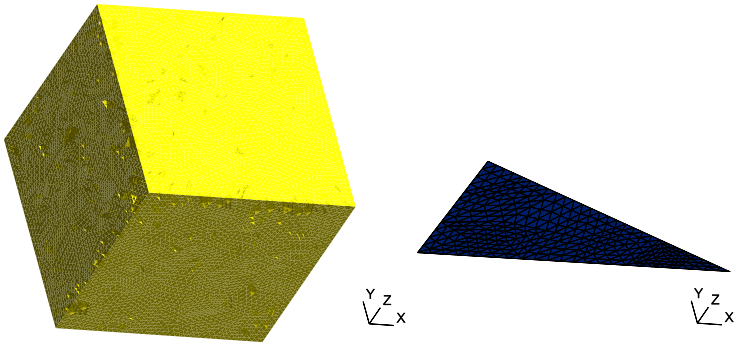
Obrázek 9.3: Příklad 2. testované sítě

3. vygenerovaná atypická síť na obrázku 9.4, která obsahuje 286315 elementů, 1 komponentu tvořenou 13663 trojúhelníky, kde většina trojúhelníků je mimo celou síť čtyřstěnů.



Obrázek 9.4: Příklad 3. testované sítě

4. vygenerovaná síť na obrázku 9.5, která obsahuje komponentu pouze uvnitř sítě čtyřstěnů, obsahuje 417888 elementů, 1 komponentu tvořenou pouze 480 trojúhelníky.



Obrázek 9.5: Příklad 4. testované sítě

5. vygenerovaná síť vypadá stejně jako 4. síť, obsahuje 267014 elementů, 1 komponentu tvořenou pouze 30 trojúhelníky.

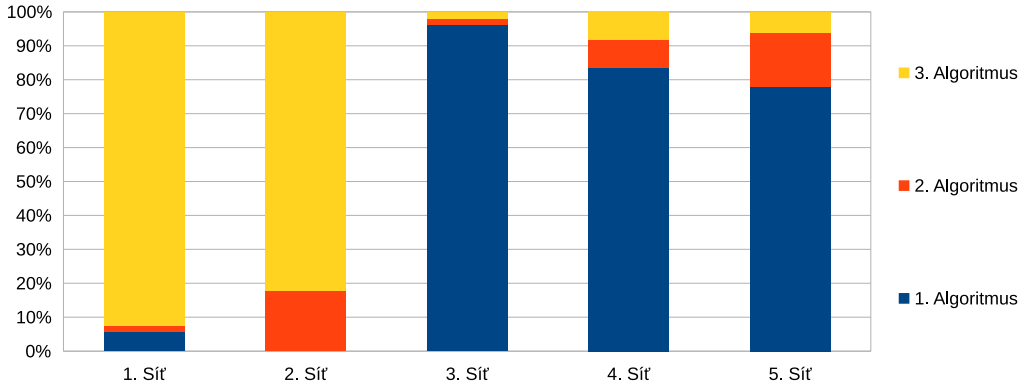
Vytvořené sítě byly testovány pro 3 algoritmy. Testovala se časová náročnost celých algoritmů, jejich inicializačních částí a výpočetních částí. Inicializační část se skládá buď z vytváření BIH Tree nebo z lineární inicializace obalových boxů (vytvoří se obalové boxy všech elementů a obalový box celé sítě čtyřstěnů).

1. algoritmus (původní z programu Flow123d) využívá pro inicializaci BIH Tree, ze kterého zjistí pro zvolený trojúhelník všechny čtyřstěny, které by s ním mohly mít průnik. Takto prochází všechny trojúhelníky v sítí.

2. algoritmus využívá pro inicializaci BIHTree i lineární inicializaci obalových boxů, podle které filtruje trojúhelníky, které jsou mimo celou síť čtyřstěnů.

3. algoritmus využívá pouze lineární inicializace obalových boxů. Hledání čtyřstěnu, se kterým by mohl mít trojúhelník průnik, je prováděno výpočtem průniku obalových boxů přes všechny čtyřstěny.

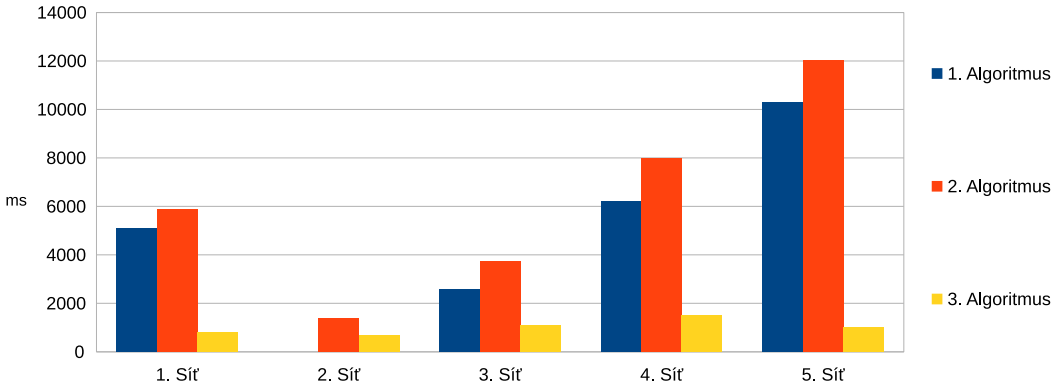
Časová náročnost výpočtů průniku sítí pro všechny algoritmy a sítě byly naměřeny 10× a časy byly zprůměrovány. Celkové časové náročnosti algoritmů jsou zobrazeny v tabulce 9.1 a v grafu v procentuálním srovnání na obrázku 9.6, časové náročnosti inicializačních částí jsou zobrazeny na obrázku 9.7 a časové náročnosti výpočetních částí jsou zobrazeny v tabulce 9.2. 1. algoritmus při výpočtu průniků pro 2. síť selhával a tak pro něj naměřené hodnoty chybí.



Obrázek 9.6: Graf celkové časové náročnosti algoritmů

Číslo sítě	1. algoritmus	2. algoritmus	3. algoritmus
1	18,768 s	10,353 s	221,745 s
2	/	3,54 s	10,654 s
3	127,46 s	6,03 s	3,76 s
4	8,71 s	8,22 s	1,75 s
5	11,31 s	12,14 s	1,09 s

Tabulka 9.1: Tabulka celkové časové náročnosti algoritmů



Obrázek 9.7: Graf časové náročnosti inicializační části algoritmů

Číslo sítě	1. algoritmus	2. algoritmus	3. algoritmus
1	13,616 s	4,459 s	220,49 s
2	/	2,15 s	9,942 s
3	124,89 s	2,30 s	2,66 s
4	2,52 s	0,24 s	0,25 s
5	1,02 s	0,21 s	0,08 s

Tabulka 9.2: Tabulka časové náročnosti výpočetní části algoritmů

Další z testů měření efektivnosti lineárního algoritmu průniku sítí 2D-3D byla provedena pro síť číslo 4 (viz obrázek 9.5) s různým počtem trojúhelníků v komponentě a různým počtem čtyřstěnů. Síť byla vytvořena s následujícími počty:

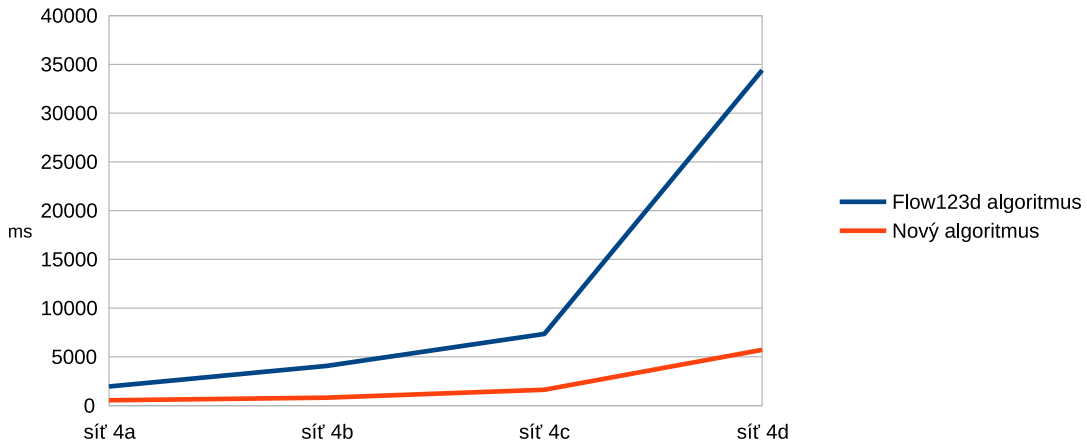
Síť 4a – 816 trojúhelníků a 47605 čtyřstěnů.

Síť 4b – 1200 trojúhelníků a 80381 čtyřstěnů.

Síť 4c – 2269 trojúhelníků a 162809 čtyřstěnů.

Síť 4d – 3282 trojúhelníků a 254099 čtyřstěnů.

Testován byl původní algoritmus z programu Flow123d a nový algoritmus, který používá lineární inicializaci obalových boxů. Naměřené hodnoty jsou znázorněny na obrázku 9.8.



Obrázek 9.8: Graf celkové časové náročnosti algoritmů pro stejnou síť s různým počtem elementů

9.1 Diskuze o rychlosti algoritmů

Nové algoritmy byly rychlejší pro sítě s různou topologií i pro sítě s různým počtem elementů. Silnou stránkou 2. a 3. algoritmu je lineární výpočet pro jednu komponentu trojúhelníků. Čím je v síti méně komponent a jednotlivé komponenty obsahují více trojúhelníků, tím budou algoritmy efektivnější. Nevýhodou pro 3. algoritmus jsou atypické sítě, které mají komponenty mimo síť čtyřstěnů, ale zároveň jsou uvnitř obalového boxu celé sítě čtyřstěnů. Pro každý trojúhelník z této komponenty se musí ověřovat průnik obalových boxů přes všechny čtyřstěny, aby se zjistilo, že neinteraguje s žádným čtyřstěnem. Dobrý příklad takového jevu lze vidět v naměřených časech pro 1. síť. Výpočetní část 2. algoritmu bude vždy efektivnější oproti 1. algoritmu. Podle typů komponent v sítí je vhodné si vybírat mezi 2. a 3. algoritmem. Uvažujeme-li, že komponenty budou vždy uvnitř celé sítě čtyřstěnů, bude 3. algoritmus výrazně efektivnější v inicializační i výpočetní části.

10 Závěr

Jedním z cílů diplomové práce bylo vytvoření optimalizovaných tříd a algoritmů pro výpočet průniku trojúhelníku s čtyřstěnem s využitím Plückerových souřadnic. Třídy a algoritmy jsem vytvořil a optimalizoval. Výhody nových algoritmů a tříd oproti již implementovaným třídám a algoritmům v programu Flow123d jsou:

- Reprezentace průniku polygonem, jehož všechny vrcholy jsou popsány barycentrickými souřadnicemi jak na trojúhelníku, tak na čtyřstěnu.
- Trasovaní polygonu bez použití dalších výpočtů.
- Zásadně menší počet součinových operací díky použití Plückerových souřadnic.
- V průměru až 59% zrychlení.

Druhým z cílů diplomové práce byl návrh, implementace a testování algoritmu pro výpočet průniků sítě trojúhelníků v síti čtyřstěnů. Algoritmus, který jsem vytvořil, umožňuje procházet síť sousedících trojúhelníků i čtyřstěnů do šířky a tím urychlit výpočet. Algoritmus jsem testoval pro 5 sítí ve dvou podobách, které se lišily inicializační částí a hledání prvotního průniku. V jistých případech bylo zrychlení o 90 % větší oproti podobným algoritmům z programu Flow123d. Pokud se použila stejná inicializační část jako v programu Flow123d, byla výpočetní část nového algoritmu vždy efektivnější. Algoritmus jsem také testoval pro jednu síť s různým počtem elementů, kde s rostoucím počtem elementů rostla i efektivnost.

Na optimalizacích algoritmu lze dále pokračovat. Mohly by se předávat vypočtené Plückerovy souřadnice mezi sousedícími elementy nebo by se Plückerovy souřadnice mohly předpočítat pro každou hranu elementu pro celou síť.

Literatura

[1] BŘEZINA, Jan. *Flow123D: Documentation of file formats and brief user manual* [online]. Verze 1.6.5. [cit. 2011-10-25]. Dostupné z: https://dev.nti.tul.cz/~brezina/flow_doc/flow123d_manual.pdf.

[2] PLATIS, Nikos; THEOHARIS, Theoharis. *Fast Ray-Tetrahedron Intersection Using Plücker Coordinates* [online]. [cit. 2003]. Dostupné z: http://users.uop.gr/~nplatis/files/PlatisTheoharisRayTetra.pdf.

[3] FRIŠ, Viktor. *Optimalizace algoritmu pro výpočet průniků simplexových výpočetních sítí.* Liberec, 2013. Bakalářská práce. Technická Univerzita Liberec. Vedoucí práce Mgr. Jan Březina, Ph.D.

[4] Learn C++. *The C++ Tutorial* [online]. [cit. 2007-05-25]. Dostupné z: http://www.learncpp.com.

[5] DORST, Leo; FONTIJNE, Daniel; MANN, Stephen. *Geometric algebra for computerscience: an object–oriented approach to geometry.* San Francisco: Morgan Kaufmann, 2007, xxxv, 626 p. Morgan Kaufmann series in computer graphics and geometric modeling. ISBN 01–237–4942–5.

[6] Wikibooks contributors. *Algorithm Implementation/Geometry/Convex hull/Monotone chain* [online]. [cit. 2014-12-26]. Wikibooks, The Free Textbook Project. Dostupné z: http://en.wikibooks.org/wiki/Algorithm_Implementation/Geometry/Convex_hull/Monotone_chain.

[7] STROUSTRUP, Bjarne. *The C programming language.* Special ed. Reading, Mass.: Addison–Wesley, c2000, x, 1019p. ISBN 02–017–0073–5.

[8] Wikibooks contributors. *Bounding interval hierarchy* [online]. [cit. 2015-04-10]. Wikipedia, The Free Encyclopedia. Dostupné z: http://en.wikipedia.org/w/index.php?title=Bounding_interval_hierarchy&oldid=651647567.

A Obsah přiloženého CD

Na přiloženém disku se nachází:

- Zkomprimovaná vývojová větev programu Flow123d s implementovanými algoritmy
- Samostatné zdrojové kódy optimalizovaných struktur a algoritmů
- Diplomová práce v elektronické podobě