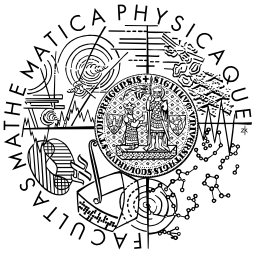


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

**BAKALÁŘSKÁ PRÁCE**

Daniel Šimek

# Algoritmy pro Minkowského součet mnohoúhelníků

Katedra algebry

Vedoucí bakalářské práce: RNDr. Zuzana Patáková, Ph.D.

Studijní program: Matematika pro informační technologie

Praha 2023

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

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

V ..... dne .....  .....

Podpis autora

Mé poděkování patří především vedoucí mé bakalářské práce, paní RNDr. Zuzaně Patákové, Ph.D., za odborné vedení, trpělivost a ochotu, se kterou se mi v průběhu zpracování celé bakalářské práce věnovala. Dále bych rád poděkoval panu RNDr. Pavlu Patákovi, Ph.D. za pomoc při instalaci knihovny CGAL. V neposlední řadě bych také rád poděkoval svým přátelům a rodině, ve kterých jsem měl po celou dobu studia plnou podporu.

Název práce: Algoritmy pro Minkowského součet mnohoúhelníků

Autor: Daniel Šimek

Katedra: Katedra algebry

Vedoucí bakalářské práce: RNDr. Zuzana Patáková, Ph.D., Katedra algebry

Abstrakt: Tato bakalářská práce se zabývá Minkowského součtem dvou nekonvexních mnohoúhelníků v rovině, konkrétně pak popisem a porovnáním dvou metod pro výpočet Minkowského součtu: rozkladové metody a konvoluční metody. V této práci jsou obě tyto metody blíže představeny, včetně potřebných definic a ilustrativních obrázků. V závěrečné kapitole jsou pak obě metody porovnány za pomoci C++ knihovny CGAL na různých vstupech.

Klíčová slova: Minkowského součet, Winding number, Nekonvexní mnohoúhelníky, Konvoluční metoda, Rozkladová metoda

Title: Algorithms for Minkowski sums of polygons

Author: Daniel Šimek

Department: Department of Algebra

Supervisor: RNDr. Zuzana Patáková, Ph.D., Department of Algebra

Abstract: This bachelor's thesis deals with the Minkowski sum of two non-convex polygons in the plane. Specifically, it focuses on describing and comparing two methods for computing the Minkowski sum: the decomposition method and the convolution method. This thesis provides a detailed presentation of both methods, including necessary definitions and illustrative images. In the final chapter, both methods are compared using the CGAL C++ library on various inputs.

Keywords: Minkowski sum, Winding number, Non-convex polygons, Convolution Method, Decomposition Method

# Obsah

# Úvod

Minkowského součet dvou množin je jedním ze základních konceptů z výpočetní geometrie, používá se v různých odvětvích, jako například robot motion planning Lengyel a kol. (1990), design řízený počítačem Kaul (1992), či detekce kolizí ve fyzikálních modelech Lee a kol. (2015).

V této práci se zaměříme především na Minkowského součet dvou nekonvexních mnohoúhelníků v rovině, celý koncept se však dá vybudovat i ve vyšších dimenzích, či pro jiné množiny bodů než pouze pro mnohoúhelníky.

Minkowského součet dvou nekonvexních mnohoúhelníků se počítá za pomoci jedné ze dvou metod: rozkladové metody a konvoluční metody. Konvoluční metoda byla včetně implementace v jazyce C++ představena například v práci Wein (2006), ze které tato práce bude vycházet.

Celý koncept Minkowského sumy nekonvexních mnohoúhelníků v této práci vysvětlíme v celkem třech kapitolách. Ve čtvrté pak na experimentálních datech vyzkoušíme rychlosti jednotlivých metod.

V první kapitole představíme základní pojmy a ukážeme postup, jak se počítá Minkowského součet, máme-li na vstupu dva konvexní mnohoúhelníky. Kapitolu doplníme kódem v Pythonu, který je implementací tohoto postupu a také jednoduchým příkladem sloužícím čtenáři k lepšímu pochopení.

V druhé kapitole popíšeme rozkladovou metodu, nastíníme, jak se dá nekonvexní mnohoúhelník rozdělit na konvexní části, a tento postup demonstrujeme na ilustrativních obrázcích a příkladech. Pro tyto již konvexní části pak využijeme poznatky z první kapitoly.

Třetí kapitola se pak bude věnovat popisu konvoluční metody. Kapitolu opět doplníme kódy v Pythonu, ilustrativními obrázky a příklady.

V poslední čtvrté kapitole pak na námi vytvořených mnohoúhelnících za pomoci C++ knihovny CGAL porovnáme obě popsané metody.

Mým příspěvkem v této práci je doplnění a uspořádání základních definic, které citované články často vynechávají, přepsání pseudokódů z citovaných článků do funkčních kódů v jazyce Python, doplnění celé teorie o zajímavé příklady a ilustrativní obrázky. V neposlední řadě jsem k problematice přispěl vygenerováním svých vlastních mnohoúhelníků, na kterých jsem s pomocí knihovny CGAL obě zmíněné metody porovnal.

# 1. Minkowského součet konvexních mnohoúhelníků

V této kapitole představíme Minkowského součet pro konvexní mnohoúhelníky v rovině a jeho algoritmus výpočtu. Demonstrujeme ho na jednoduchých příkladech.

Nejprve si obecně zadefinujeme Minkowského součet dvou množin.

**Definice 1** (Minkowského součet)**.** *Mějme dvě množiny $A$, $B \subset \mathbb{R}^n$. Jejich* Minkowského součet*, značíme $A \oplus B$, je množina $\{a+b : a \in A, b \in B\}$, kde jednotlivé body sčítáme po souřadnicích.*

Nyní ještě potřebujeme zadefinovat základní pojmy, které budeme po celou dobu této práce používat. Zadefinujeme si je pro $\mathbb{R}^2$, nicméně se dají rozšířit i do vyšších dimenzí.

**Definice 2.** *Množinu $M \subset \mathbb{R}^2$ nazveme* konvexní*, jestliže pro všechny body $m, n$ ležící v $M$ a pro všechny $t$ z intervalu* [0,1] *platí, že $tm + (1-t)n$ leží v $M$.*

**Definice 3.** Konvexní obal *množiny $M \subset \mathbb{R}^2$, značíme $conv(M)$, je množina všech bodů $x$ z $\mathbb{R}^2$, pro které existuje konečně mnoho bodů $m_1, m_2, ..., m_k \in M$ a $a_1, a_2, ..., a_k$ z intervalu* [0,1]*, $\sum_{i=1}^{k} a_i = 1$ tak, že platí $x = \sum_{i=1}^{k} a_i m_i$.*

**Definice 4.** *Buď $P$ konvexní obal neprázdné množiny $M \subset \mathbb{R}^2$, pak bod $v \in P$ nazveme* extremální*, pokud platí, že je-li $v = tm + (1-t)n$ pro $t$ z intervalu* (0,1) *a $m$,$n \in P$, pak nutně $m = n$. Množinu extremálních bodů z $P$ budeme značit $ext(P)$.*

Tvrzení z knihy (Matousek, 2013, str. 87) nám navíc říká důležitou vlastnost extremálních bodů, a to že $conv(ext(P)) = P$.

**Definice 5.** Konvexním mnohoúhelníkem *$P$ rozumíme konvexní obal libovolné konečné množiny $M \subset \mathbb{R}^2$ obsahující alespoň tři body, které neleží na jedné přímce. Množinu extremálních bodů z $P$ nazveme* množina vrcholů*, značíme $V$, a její prvky* vrcholy konvexního mnohoúhelníku *$P$.*

Pro další teorii musíme nyní množinu vrcholů vhodně seřadit. Mějme tedy nějaký konvexní mnohoúhelník $P$ a jeho množinu vrcholů $V$ s $n$ vrcholy. Zvolíme libovolný bod $A \in P \setminus V$ a vrchol $v$ s nejmenší $y$-ovou souřadnicí (je-li takových vrcholů více, pak z nich vybereme vrchol s nejmenší $x$-ovou souřadnicí). Tento vrchol $v$ označíme $v_1$ a vezmeme si polopřímku $R = [A,v_1)$, kterou budeme rotovat okolo bodu $A$ proti směru hodinových ručiček, dokud neprotneme další vrchol. Ten označíme $v_2$ a rotujeme dál. Takto pokračujeme dokud se nedostaneme zpět na původní pozici. Tímto způsobem dostaneme vhodně seřazené vrcholy $v_1$,$v_2$,...,$v_n$. Úsečkám $v_i$,$v_{i+1}$, pro $i \in \{1,...,n\}$ budeme říkat hrany. Zde je nutné poznamenat, že $v_{n+1} = v_1$.

Máme-li tedy dva konvexní mnohoúhelníky $P$ a $Q \subset \mathbb{R}^2$, pak je podle Definice 1 jejich Minkowského součet definován jako

$$P \oplus Q = \{p + q : p \in P, q \in Q\}.$$

Problém nastává ve chvíli, kdy chceme nějaký takovýto součet spočítat. Jelikož tyto mnohoúhelníky obsahují nekonečně mnoho bodů, tak nemůžeme spočítat všechny jejich kombinace. Ukazuje se však, že stačí brát pouze množinu vrcholů daných mnohoúhelníků a Minkowského součet spočítat jen pro ně, tedy že platí následující věta.

**Věta 6.** *Buď $P$ a $Q$ dva konvexní mnohoúhelníky, $V$ množina vrcholů mnohoúhelníku $P$ a $U$ množina vrcholů mnohoúhelníku $Q$, pak platí $P \oplus Q = conv(V \oplus U)$.*

*Důkaz.* Prvně si uvědomíme, že $P = conv(V)$ a $Q = conv(U)$, dokazujeme tedy rovnost $conv(V) \oplus conv(U) = conv(V \oplus U)$. S využitím poznatků z (Fradelizi a kol., 2018, str. 7, Lemma 2.1) dokážeme dvě inkluze.
"$\supseteq$": Nechť $x \in conv(V \oplus U)$, pak se podle definice dá napsat jako $\sum_{i=1}^{k} a_i w_i$, kde $\sum_{i=1}^{k} a_i = 1$ a všechny $a_i$ jsou z intervalu [0,1]. Navíc $w_i \in V \oplus U$, tedy $w_i = v_i + u_i$, pro $v_i \in V$ a $u_i \in U$. Tedy po rozepsání $\sum_{i=1}^{k} a_i w_i = \sum_{i=1}^{k} a_i(v_i + u_i) = \sum_{i=1}^{k} a_i v_i + \sum_{i=1}^{k} a_i u_i \in conv(V) \oplus conv(U)$.

"$\subseteq$": Nechť $x \in conv(V) \oplus conv(U)$, pak $x = \sum_{i=1}^{k} a_i v_i + \sum_{j=1}^{l} b_j u_j$, kde $v_i \in V$, $u_j \in U$, $\sum_{i=1}^{k} a_i = 1$ a $\sum_{j=1}^{l} b_j = 1$ a všechny $a_i$ i $b_j$ jsou z intervalu [0,1]. Navíc jsou obě sumy konečné, můžeme tedy napsat, že $x = \sum_{j=1}^{l} b_j \sum_{i=1}^{k} a_i v_i + \sum_{i=1}^{k} a_i \sum_{j=1}^{l} b_j u_i = \sum_{i=1}^{k} \sum_{j=1}^{l} a_i b_j v_i + \sum_{i=1}^{k} \sum_{j=1}^{l} a_i b_j u_j = \sum_{i=1}^{k} \sum_{j=1}^{l} a_i b_j (v_i + u_j) \in conv(V \oplus U)$.

□

**Důsledek 7.** *Minkowského součet dvou konvexních mnohoúhelníků je opět konvexní mnohoúhelník.*

Pro zjednodušení budeme odteď fakt, že $P = conv(V)$, značit pouze jako $P = V$.

## 1.1 Algoritmus výpočtu Minkowského součtu dvou konvexních mnohoúhelníků

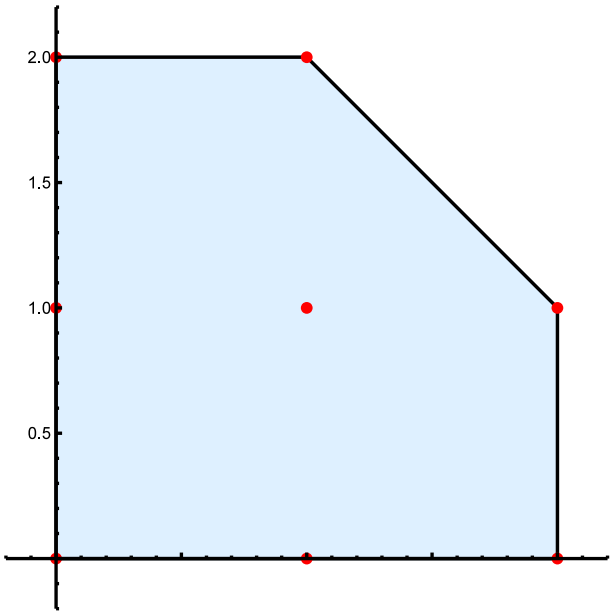
Nyní bude předveden algoritmus pro výpočet Minkowského součtu. Na vstupu máme dva mnohoúhelníky reprezentované množinou svých vrcholů. Na výstupu pak dostaneme množinu bodů, jejichž konvexní obal nám reprezentuje výsledný Minkowského součet.

Z Věty 6 nás intuitivně napadne sečíst každý vrchol P s každým vrcholem Q, a přesně to dělá tento kód (Python).

```
def Minkowskeho_soucet(P, Q):
    kombinace = []
    for p in P:
        for q in Q:
            kombinace.append((p[0] + q[0], p[1] + q[1]))
    return kombinace
```

Například pro mnohoúhelníky $P = [(0,0),(1,0),(0,1)]$ a $Q = [(0,0),(1,0),(1,1),(0,1)]$ dostaneme:

```
[(0, 0), (1, 0), (1, 1), (0, 1), (1, 0), (2, 0), (2, 1),
(1, 1), (0, 1), (1, 1), (1, 2), (0, 2)]
```




Obrázek 1.1: Grafický výsledek (Wolfram Mathematica). Minkowského součet je celá modrá plocha, včetně červených bodů a jejich spojnic.

Můžeme si všimnout, že se zde několik bodů opakuje a dokonce, jak vidíme z Obrázku 1.1, ne všechny tyto body (červené) jsou vrcholy nově vzniklého mnohoúhelníku. Nabízí se tedy otázka, zda neexistuje efektivnější algoritmus, který by spočítal pouze body určující výsledný mnohoúhelník (tedy jeho vrcholy). A pokud ano, kolik nejméně takových bodů potřebujeme spočítat. Částečnou odpověď na tuto otázku nám dává následující věta:

**Věta 8.** *Buď $P$ a $Q$ dva konvexní mnohoúhelníky s $n$ a $m$ vrcholy, pak Minkowského součet $P \oplus Q$ je konvexní mnohoúhelník s nejvýše $n+m$ vrcholy.*

*Důkaz.* To, že Minkowského součet bude opět konvexní, plyne z Důsledku 7, zbytek důkazu je popsán v knize de Berg a kol. (2008, str. 292, Věta 13.5). Je zde však ukázán pro $n$ a $m$ hran, kdežto my máme ve znění věty $n$ a $m$ vrcholů. Uvědomíme-li si však, že počet hran mnohoúhelníku odpovídá počtu jeho vrcholů, tak můžeme v této větě tyto dva pojmy zaměnit se stejným výsledkem. □

Dále je v knize de Berg a kol. (2008, str. 295) popsán a v pseudokódu předveden efektivnější algoritmus, počítající Minkowského součet dvou konvexních mnohoúhelníků.

Zde je přepsaný do jazyka Python [1].

[1] Předpokládá se, že mnohoúhelníky vstupující do tohoto algoritmu začínají vrcholem s nejmenší souřadnicí $y$ (popřípadě souřadnicí $x$, je-li více takových vrcholů).

```
import math
def uhel(p, q):
    #math.atan2 vraci uhel (rad), ktery vstupni vektor svira s
    #kladnou poloosou x
    pom=math.atan2(q[1]-p[1], q[0]-p[0])
    if (pom<0): return 2*math.pi+pom
        #je-li uhel vetsi nez pi, math.atan2 vrati uhel s
        # kladnou poloosou x ve smeru hodinovych rucicek,
        #my vsak chceme uhel proti smeru
    else: return pom
def minkowski_efekt(P, Q):
    n = len(P), m = len(Q)
    i, j, k, l = 0, 0, 0, 0
    vysledek = []
    while (i < n and j <= m) or (i <= n and j < m):
        vysledek.append((P[i%n][0]+Q[j%m][0], P[i%n][1]+Q[j%m][1]))
        uhelP = uhel(P[i%n], P[(i+1)%n])
        uhelQ = uhel(Q[j%m], Q[(j+1)%m])
        if i>=n: k=2*math.pi
        if j>=m: l=2*math.pi
        #kdyz u nektereho mnohouhelniku dojdeme kolem dokola,
        #tak chceme aby i uhel byl o kolo (2pi) vetsi
        if uhelP+k < uhelQ+l:
            i+=1
        elif uhelP+k > uhelQ+l:
            j+=1
        else:
            i+=1
            j+=1
    return vysledek
```

Pustíme-li tento kód opět na mnohoúhelníky $P = [(0,0),(1,0),(0,1)]$ a $Q = [(0,0),(1,0),(1,1),(0,1)]$ dostaneme:

```
[(0, 0), (2, 0), (2, 1), (1, 2), (0, 2)]
```

Je vidět, že tento efektivnější algoritmus nám pro tento příklad dá na výstupu pouze vrcholy nově vzniklého mnohoúhelníku.

## 1.2 Další příklady

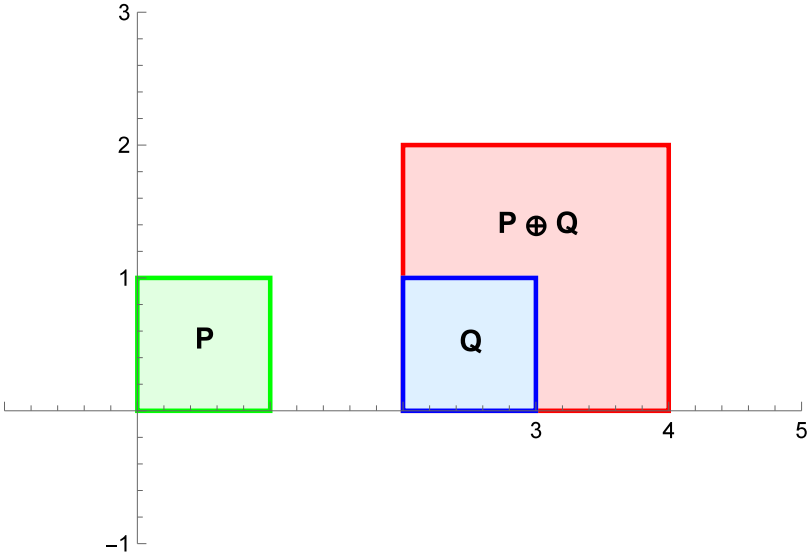
Pro lepší pochopení algoritmu počítajícího Minkowského součet dvou konvexních mnohoúhelníků zde ještě předvedeme dva příklady. Na prvním příkladě ukážeme, že výsledný mnohoúhelník, může mít stejný počet vrcholů, jako mnohoúhelníky vstupující do algoritmu. A na druhém příkladě si ukážeme opačný jev, kdy počet vrcholů vzniklého mnohoúhelníku bude maximální, tedy $m + n$.

***Příklad 1:*** Mějme $P = [(0,0),(1,0),(1,1),(0,1)]$ a $Q = [(2,0),(3,0),(3,1),(2,1)]$. Spočítejte $P \oplus Q$.

***Řešení:***

| Krok | i | j | vysledek [] | $P[i]+Q[j]$ | porovnání úhlů |
|---|---|---|---|---|---|
| 1 | 0 | 0 | [] | (0,0) + (2,0) = (2,0) | $uhelP = uhelQ$ $i++, j++$ |
| 2 | 1 | 1 | [(2,0)] | (1,0) + (3,0) = (4,0) | $uhelP = uhelQ$ $i++, j++$ |
| 3 | 2 | 2 | [(2,0), (4,0)] | (1,1) + (3,1) = (4,2) | $uhelP = uhelQ$ $i++, j++$ |
| 4 | 3 | 3 | [(2,0), (4,0), (4,2)] | (0,1) + (2,1) = (2,2) | $uhelP = uhelQ$ $i++, j++$ |

Podmínky *while* cyklu již nejsou splněny, $vysledek = [(2,0), (4,0), (4,2), (2,2)]$

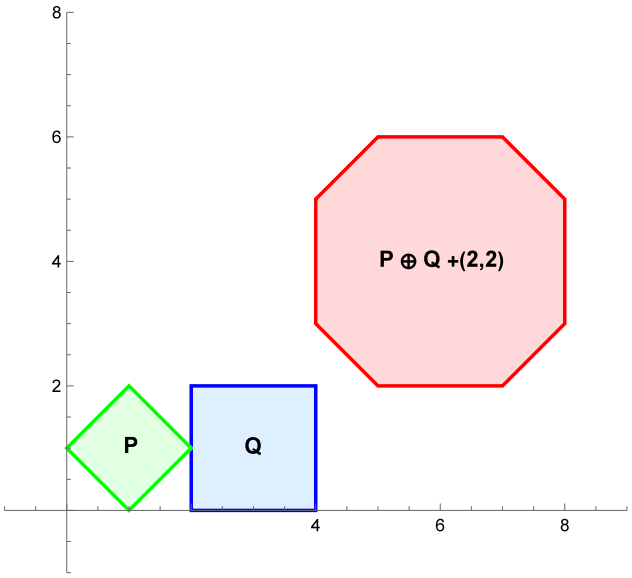



***Příklad 2:*** Mějme $P = [(1,0),(2,1),(1,2),(0,1)]$ a $Q = [(2,0),(4,0),(4,2),(2,2)]$. Spočítejte $P \oplus Q$.

***Řešení:***

| Krok | i | j | vysledek [] | $P[i]+Q[j]$ | porovnání úhlů |
|---|---|---|---|---|---|
| 1 | 0 | 0 | [] | (1,0) + (2,0) = (3,0) | $uhelP > uhelQ$ $j++$ |
| 2 | 0 | 1 | [(3,0)] | (1,0) + (4,0) = (5,0) | $uhelP < uhelQ$ $i++$ |
| 3 | 1 | 1 | [(3,0), (5,0)] | (2,1) + (4,0) = (6,1) | $uhelP > uhelQ$ $j++$ |
| 4 | 1 | 2 | [(3,0), (5,0), (6,1)] | (2,1) + (4,2) = (6,3) | $uhelP < uhelQ$ $i++$ |
| 5 | 2 | 2 | [(3,0), (5,0), (6,1), (6,3)] | (1,2) + (4,2) = (5,4) | $uhelP > uhelQ$ $j++$ |
| 6 | 2 | 3 | [(3,0), (5,0), (6,1), (6,3), (5,4)] | (1,2) + (2,2) = (3,4) | $uhelP < uhelQ$ $i++$ |
| 7 | 3 | 3 | [(3,0), (5,0), (6,1), (6,3), (5,4), (3,4)] | (0,1) + (2,2) = (2,3) | $uhelP > uhelQ$ $j++$ |
| 8 | 3 | 4 | [(3,0), (5,0), (6,1), (6,3), (5,4), (3,4), (2,3)] | (0,1) + (2,0) = (2,1) | $uhelP < uhelQ$ $i++$ |

Podmínky *while* cyklu již nejsou splněny,
$vysledek = [(3,0),(5,0),(6,1),(6,3),(5,4),(3,4),(2,3),(2,1)]$




Obrázek 1.2: Výsledný mnohoúhelník je posunut o vektor (2,2), aby byl lépe vidět.

## 1.3 Nekonvexní mnohoúhelník

Než pokročíme dále, musíme si ještě zadefinovat pár dalších definic, které využijeme v dalších kapitolách.

**Definice 9.** Jednoduchým mnohoúhelníkem $P$ *budeme rozumět uzavřenou omezenou část* $\mathbb{R}^2$ *ohraničenou uzavřenou lomenou čarou s konečně mnoha zlomy* [2]*, pro kterou navíc platí, že se nikde neprotíná, a pro body, které tuto lomenou čáru určují, platí, že žádné tři po sobě jdoucí neleží na jedné přímce. Těmto bodům budeme říkat* vrcholy *a množinu těchto bodů označíme* $V$*. Úsečkám, ze kterých je tato lomená čára složená, pak budeme říkat* hrany*. Množině bodů z* $P \setminus V$ *takových, že neleží na žádné hraně budeme říkat* vnitřek mnohoúhelníku $P$ *a budeme ji značit* $in(P)$*. Jednoduchý mnohoúhelník pak budeme reprezentovat jeho vrcholy v pořadí v jakém leží na uzavřené lomené čáře.*

**Lemma 10.** *Každý konvexní mnohoúhelník je jednoduchý.*

*Důkaz.* Buď $P$ nějaký konvexní mnohoúhelník s hranami $v_1v_2, v_2v_3,...,v_nv_1$. Tyto hrany tvoří uzavřenou lomenou čáru, která se nikde neprotíná a žádné tři vrcholy neleží na jedné přímce.

□

**Definice 11.** Nekonvexním mnohoúhelníkem *nazveme takový jednoduchý mnohoúhelník, který není konvexní.*

[2] To, že tato lomená čára dělí rovinu na omezenou a neomezenou část platí z Jordanovy věty Hales (2007).

Jednoduché mnohoúhelníky budou právě ty, se kterými budeme Minkowského součet počítat, výsledný mnohoúhelník však už jednoduchý být nemusí, viz Obrázek 4.7. Takovému mnohoúhelníku budeme říkat mnohoúhelník s dírami.

**Definice 12.** *Buď $P$ jednoduchý mnohoúhelník a $D_i \subset in(P)$ konečně mnoho jednoduchých mnohoúhelníků, pak množinu $P_D = P \setminus (\bigcup_i D_i)$ nazveme* mnohoúhelník s dírami.

Pro doplnění zde ještě uveďme vrchní meze počtu vrcholů výsledného Minkowského součtu. Z Věty 8 víme, že mnohoúhelník vzniklý z Minkowského součtu dvou konvexních mnohoúhelníků bude mít nejvíce $m + n$ vrcholů. Z Kaul (1991) a Dobkin a kol. (1990) pak dostaneme, že sečtením jednoho konvexního a jednoho nekonvexního mnohoúhelníku vznikne mnohoúhelník s nejvýše $O(mn)$ vrcholy a sečtením dvou nekonvexních nám může dát mnohoúhelník s až $O(m^2n^2)$ vrcholy, kde $m$ je počet vrcholů prvního mnohoúhelníku a $n$ je počet vrcholů druhého. V Kaul (1991) je ukázán i příklad dvou nekonvexních mnohoúhelníků, jejichž Minkowského součet nabývá vrchní meze počtu vrcholů.

# 2. Rozkladová metoda

V minulé kapitole jsme popsali a ukázali efektivní algoritmus na výpočet Minkowského součtu dvou konvexních mnohoúhelníků. V této kapitole na tu předchozí navážeme a ukážeme, jak spočítat Minkowského součet dvou mnohoúhelníků, kde alespoň jeden není konvexní, pomocí rozkladové metody. Tato metoda využívá, stejně jako nemalá část složitějších matematických problémů, tzv. princip *Divide Et Impera*. Neboli rozdělení celého problému do menších částí, které jsme schopni lépe, či rychleji spočítat.

## 2.1 Popis metody

Prvním krokem je rozložení nekonvexních mnohoúhelníků na konvexní části pomocí nějakého rozkladového algoritmu. Pro tyto již konvexní části poté využijeme poznatky z první kapitoly a spočítáme pro každou z nich Minkowského součet s druhým mnohoúhelníkem, popřípadě se všemi jeho konvexními částmi, byl-li i druhý mnohoúhelník nekonvexní. Nakonec pak výsledné množiny sjednotíme, čímž dostaneme Minkowského součet původních mnohoúhelníků.

## 2.2 Rozkladové algoritmy

**Definice 13.** *Mějme rozkladový algoritmus a vstupní vrcholy nekonvexního mnohoúhelníku. Vznikne-li nám během algoritmu nový vrchol, který nebyl součástí vstupních vrcholů, pak takový vrchol nazveme* Steinerův bod.

Obecně existuje velká škála různých rozkladových algoritmů, my v této práci představíme tři z nich. Blíže pak popíšeme dva, které demonstrujeme na jednoduchých příkladech. Budeme při tom vycházet z popisu těchto algoritmů z (Agarwal a kol., 2002, str. 44-46). První algoritmus bude bez Steinerových bodů a druhý bude se Steinerovými body.

### 2.2.1 Triangulace

V prvním kroku tento algoritmus nalezne dva vrcholy, jejichž spojnice leží uvnitř vstupního mnohoúhelníku a rozdělí tento mnohoúhelník podle této spojnice. Rekurzivně se pak tento algoritmus pustí na nově vzniklé mnohoúhelníky. Celý proces končí ve chvíli, kdy už má nově vzniklý mnohoúhelník pouze tři vrcholy, tedy ve chvíli, kdy neexistuje žádná spojnice mezi vrcholy, která by ležela uvnitř tohoto mnohoúhelníku. Neboli algoritmus skončí, když je nový mnohoúhelník trojúhelník.

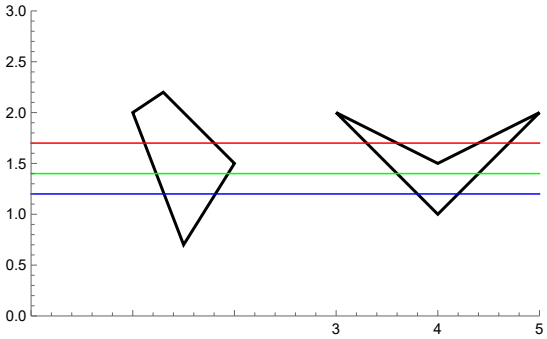
### 2.2.2 Angle bisector

Tento algoritmus vezme postupně všechny nekonvexní vnitřní úhly vstupního mnohoúhelníku. Pro každý z nich vezme jeho osu a z příslušného vrcholu narýsuje ve směru této osy úsečku. Ta skončí na hraně původního mnohoúhelníku, na již

narýsované úsečce (tím vznikne Steinerův bod), nebo v původním vrcholu. Tyto úsečky tak rozdělí mnohoúhelník na konvexní části.

### 2.2.3 Greene

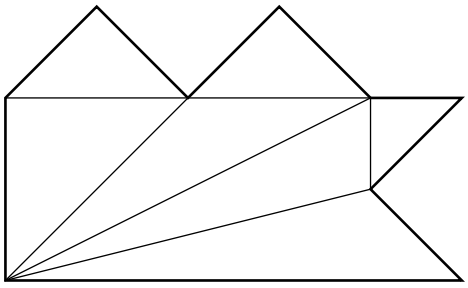
V Kapitole 4, ve které budeme jednotlivé metody porovnávat, ještě využijeme Greene dekompoziční algoritmus, představený v Greene (1983). Ten rozděluje nekonvexní mnohoúhelníky na *y-monotónní*. Mnohoúhelník nazveme $y$-monotónní, jestliže jakákoli přímka kolmá na osu $y$ protne tento mnohoúhelník v nejvýše dvou bodech z lomené čáry, která tento mnohoúhelník určuje.



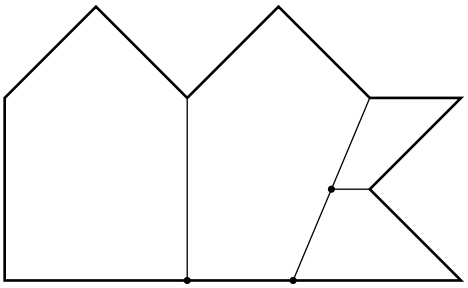

Obrázek 2.1: Mnohoúhelník vlevo je $y$-monotónní, mnohoúhelník napravo není, protože červená přímka ho protíná ve čtyřech bodech.

## 2.3 Ukázky rozkladů

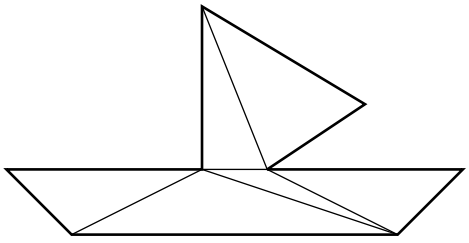
Nyní ukážeme použití výše zmíněných algoritmů na různých nekonvexních mnohoúhelnících. Rozdělení není vždy jednoznačné a záleží na konkrétní implementaci algoritmu (v jakém vrcholu začneme, které dva vrcholy spojíme, ...).



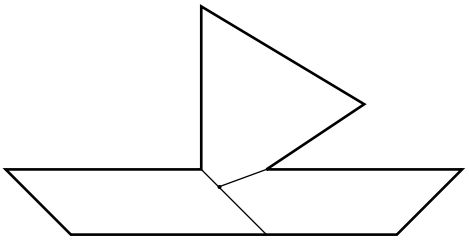
Obrázek 2.2: Triangulace



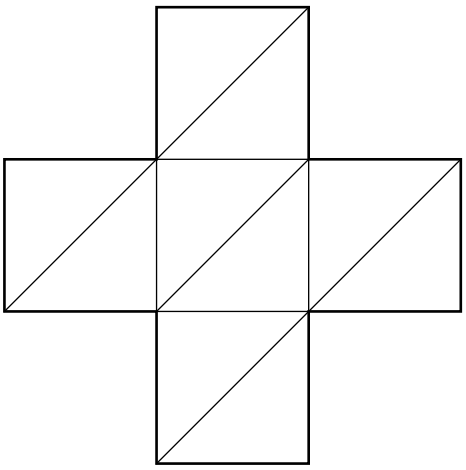
Obrázek 2.3: Angle bisector



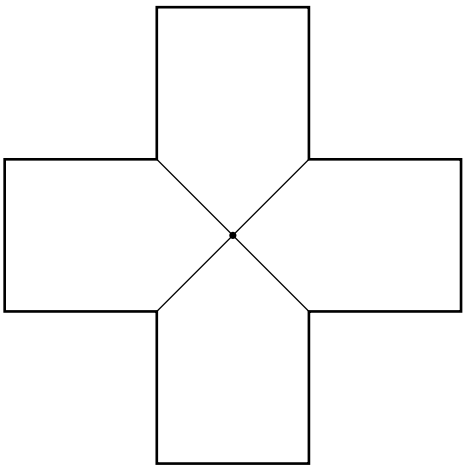
Obrázek 2.4: Triangulace



Obrázek 2.5: Angle bisector



Obrázek 2.6: Triangulace

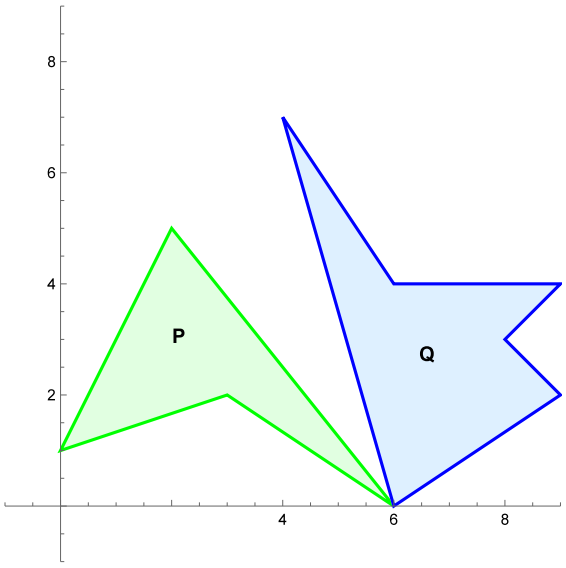


Obrázek 2.7: Angle bisector

## 2.4 Příklad na rozkladovou metodu

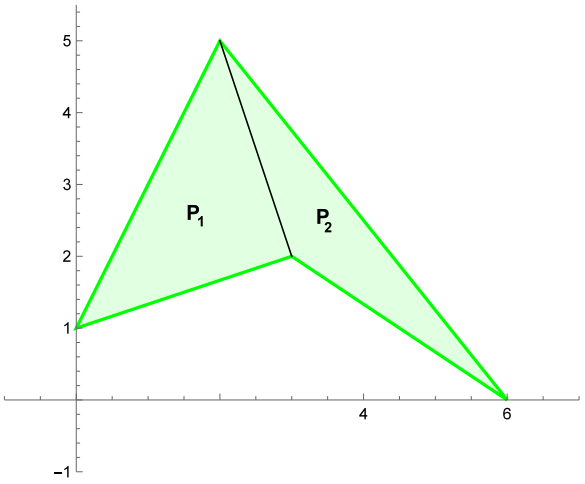
Pro ještě lepší pochopení uvedeme na závěr této kapitoly jeden příklad na Minkowského součet dvou nekonvexních mnohoúhelníků pomocí rozkladové metody.

***Příklad 3:*** Mějme $P = [(0,1),(3,2),(6,0),(2,5)]$ a $Q = [(6,0),(9,2),(8,3),(9,4),(6,4),(4,7)]$. Spočítejte $P \oplus Q$.



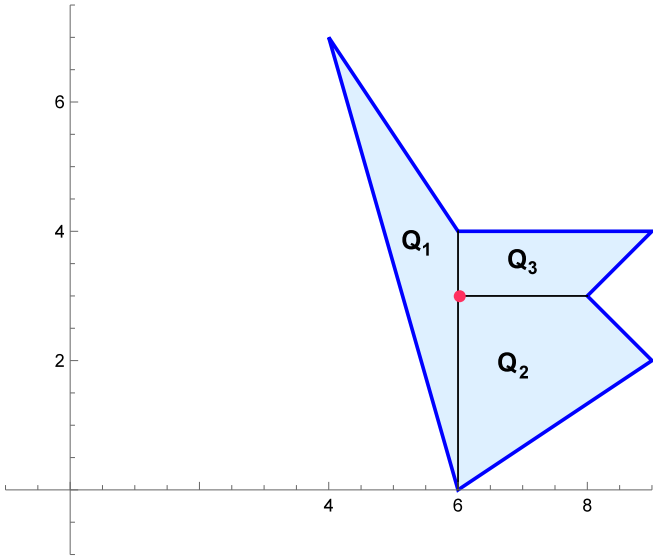

Obrázek 2.8: Vstupní mnohoúhelníky (Wolfram Mathematica).

Vidíme, že oba mnohoúhelníky jsou nekonvexní, musíme je tedy rozdělit na konvexní části. Začneme s mnohoúhelníkem $P$, na který použijeme triangulační algoritmus. Dostaneme dva již konvexní mnohoúhelníky $P_1 = [(0,1),(3,2),(2,5)]$ a $P_2 = [(6,0),(2,5),(3,2)]$.




Obrázek 2.9: Rozklad mnohoúhelníku $P$ pomocí triangulace (Wolfram Mathematica).

Mnohoúhelník $Q$ rozložíme tak, aby nám vznikl Steinerův bod (červený). Jedná se téměř o algoritmus Angle bisector, jen s tím rozdílem, že u prvního úhlu nebereme jeho osu, a to z důvodu hezčích čísel pro další počítání. Dostaneme tak tyto tři konvexní mnohoúhelníky $Q_1 = [(6,0),(6,4),(4,7)], Q_2 = [(6,0),(9,2),(8,3),(6,3)]$ a $Q_3 = [(6,3),(8,3),(9,4),(6,4)]$.

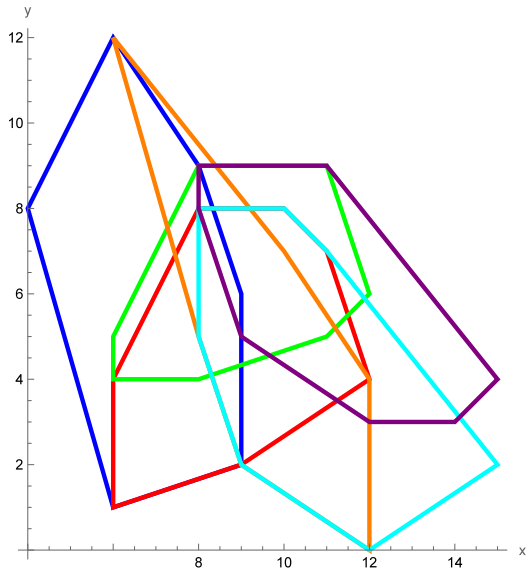



Obrázek 2.10: Rozklad mnohoúhelníku $Q$ (Wolfram Mathematica).

Nyní spočítáme $P_i \oplus Q_j$ pro $i = 1, 2$ a $j = 1, 2, 3$ a dostaneme:

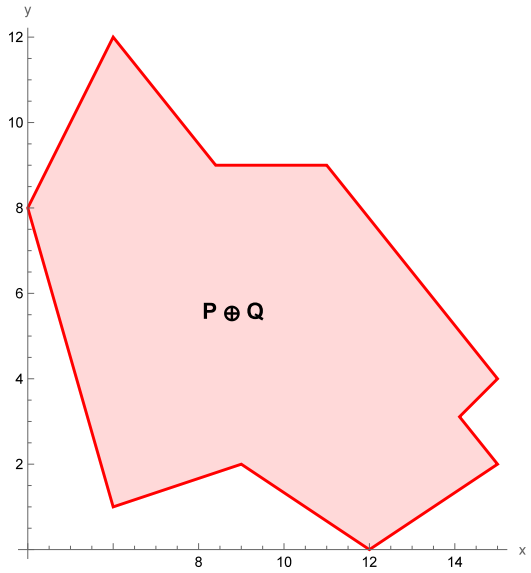
$V_1 = P_1 \oplus Q_1 = [(6,1),(9,2),(9,6),(8,9),(6,12),(4,8)]$ (Modrý)
$V_2 = P_1 \oplus Q_2 = [(6,1),(9,2),(12,4),(11,7),(10,8),(8,8),(6,4)]$ (Červený)
$V_3 = P_1 \oplus Q_3 = [(6,4),(8,4),(11,5),(12,6),(11,9),(8,9),(6,5)]$ (Zelený)
$V_4 = P_2 \oplus Q_1 = [(12,0),(12,4),(10,7),(6,12),(8,5),(9,2)]$ (Žlutý)
$V_5 = P_2 \oplus Q_2 = [(12,0),(15,2),(11,7),(10,8),(8,8),(8,5),(9,2)]$ (Cyan)
$V_6 = P_2 \oplus Q_3 = [(12,3),(14,3),(15,4),(11,9),(8,9),(8,8),(9,5)]$ (Fialový)

Necháme-li si nyní vykreslit všechny mezivýsledky $V_k$, tak dostaneme:




Obrázek 2.11: Jednotlivé mnohoúhelníky $V_k$, pro $k = 1, 2,...,6$, pro větší přehlednost nemají vyplněné vnitřky, které tam ovšem samozřejmě patří. (Wolfram Mathematica).

Sjednocením těchto mnohoúhelníků dostaneme výsledek, tedy $P \oplus Q = [(11, 9), (8.4, 9), (6, 12), (4, 8), (6, 1), (9, 2), (12, 0)(15, 2), (14.1, 3.1), (15, 4)]$.




Obrázek 2.12: Výsledný Minkowského součet pomocí rozkladové metody.

# 3. Konvoluční metoda

Další možností, jak počítat Minkowského součet dvou nekonvexních mnohoúhelníků, je konvoluční metoda. Ta se skládá ze dvou částí, které si postupně popíšeme. V této kapitole budeme vycházet z Wein (2006), kde je tato metoda popsána. V průběhu kapitoly navíc konvoluční metodu demonstrujeme na jednoduchém příkladu. Připomeňme, že jako vstupní mnohoúhelníky připouštíme pouze jednoduché mnohoúhelníky, nicméně výstupní mnohoúhelník může být mnohoúhelník s dírami. Myšlenka za touto metodou vychází z Guibas a kol. (1983). Idea je, že konvoluce dvou mnohoúhelníků tvoří nadmnožinu hranám jejich Minkowského součtu a Minkowského součet je pak množina bodů s nenulovým winding number vzhledem ke konvolučním cyklům. Jak je však zmíněno v Baram a kol. (2018), tato skutečnost nebyla nikde zcela dokázána. O tom svědčí i zpráva, kterou jsem dostal od Prof. Dana Halperina z Tel Avivské univerzity. Cituji: „*I refrain from using the convolution method for non-convex polygons,as I have never seen a satisfactory description cum rigorous proof of it, nor a fully robust implementation of it*“. Proto bude mít tato kapitola spíše popisný charakter, využívající poznatky z citovaných článků.

## 3.1 Konvoluční cykly

**Definice 14.** *Mějme nenulový vektor $u = (a, b)$. Pak definujeme úhel $\Phi$, který svírá vektor $u$ s kladnou poloosou $x$ následovně:*

$$\Phi = \begin{cases} arctg(\frac{b}{a}) & a > 0 \text{ a zároveň } b \geq 0. \\ \frac{\pi}{2} & a = 0 \text{ a zároveň } b > 0 \\ arctg(\frac{b}{a}) + \pi & a < 0 \\ \frac{3\pi}{2} & a = 0 \text{ a zároveň } b < 0 \\ arctg(\frac{b}{a}) + 2\pi & a > 0 \text{ a zároveň } b < 0 \end{cases}$$

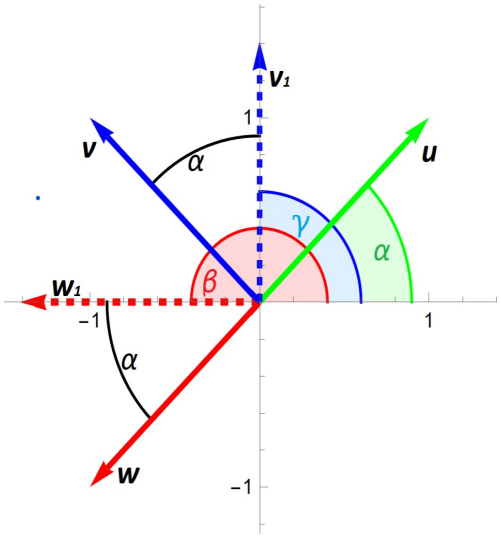
**Definice 15.** *Mějme nenulové vektory $u, v$ a $w$. Buď $\alpha$ úhel, který vektor $u$ svírá s kladnou poloosou $x$ a nechť $v_1$ vznikne rotací $v$ o úhel $-\alpha$ a obdobně $w_1$ vznikne rotací vektoru $w$ o stejný úhel $-\alpha$. Buď nyní $\beta$ úhel, který svírá $w_1$ s kladnou poloosou $x$ a $\gamma$ úhel, který svírá $v_1$ s kladnou poloosou $x$. Pak řekneme, že* vektor $v$ leží mezi vektory $u$ *a* $w$*, pokud $\gamma$ leží v uzavřeném intervalu $[0, \beta]$* [1].

**Definice 16.** *Mějme mnohoúhelníky $P$ a $Q$ s vrcholy $p_1$,$p_2$,...$p_n$ a $q_1$,$q_2$,...$q_m$* [2]. Konvolucí *$C$ mnohoúhelníků $P$ a $Q$, značíme $P * Q$, rozumíme množinu orientovaných úseček tvaru $\overrightarrow{(p_i + q_j), (p_i + q_{j+1})}$, kde vektor $\overrightarrow{(p_i, p_{i+1})}$ leží mezi $\overrightarrow{(q_{j-1}, q_j)}$ a $\overrightarrow{(q_j, q_{j+1})}$ a symetricky z orientovaných úseček tvaru $\overrightarrow{(p_i + q_j), (p_{i+1} + g_j)}$, kde vektor $\overrightarrow{(q_j, q_{j+1})}$ leží mezi $\overrightarrow{(p_{i-1}, p_i)}$ a $\overrightarrow{(p_i, p_{i+1})}$, tyto orientované úsečky dohromady tvoří uzavřené křivky, kterým budeme říkat* konvoluční cykly [3].

[1] Povolíme-li pouze polouzavřený interval, jak je popsáno ve Wein (2006), tedy že pokud bude mít vektor $v$ i $w$ stejný směr, tak řekneme, že $v$ neleží mezi $u$ a $w$, pak nám konvoluční metoda například pro $P = [(0,0), (1,0), (0,1)]$ a $Q = [(2,0), (3,0), (3,1), (2,1)]$ nebude fungovat.

[2] Platí, že $p_{n+1} = p_1$ a $q_{m+1} = q_1$.

[3] Tato teorie opět vychází z Guibas a kol. (1983).




Obrázek 3.1: Ukázka, kdy vektor $v$ leží mezi vektory $u$ a $w$.

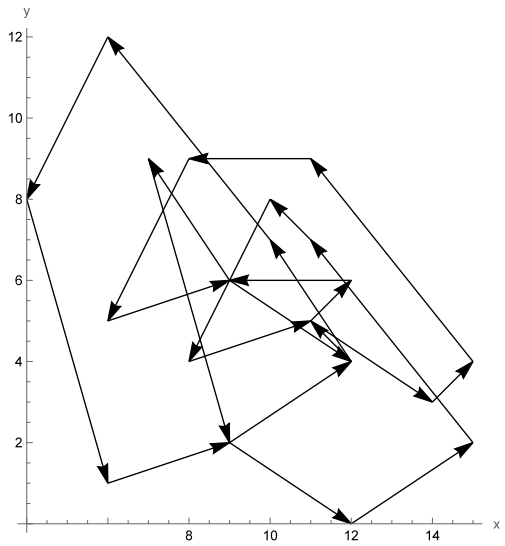
Předešlá definice konvoluce a konvolučních cyklů dvou mnohoúhelníků je překladem z anglické verze (Wein, 2006, str. 3), kde je i pomocí pseudokódu popsán algoritmus, který tuto konvoluci počítá. My si ho zde ukážeme přepsaný do Pythonu.

```
import numpy as np
def uhel(p):
    pom = np.arctan2(p[1], p[0])
    if pom < 0:
        return 2 * np.pi + pom
    else: return pom
def mezi(u, v, w):
    w = uhel(w)
    u, v = uhel(u) - w, uhel(v) - w
    if u < 0:
        u = u + 2 * np.pi
    if v < 0:
        v = v + 2 * np.pi
    if u <= v:
        return True
    if v==0: return True
    else: return False
def cykly(P, Q):
    n=len(P)
    m=len(Q)
    for i in range(n):
        for j in range(m):
            a, b, c = P[(i-1)], P[i], P[(i+1)%n]
            d, e, f = Q[(j-1)], Q[j], Q[(j+1)%m]
            if mezi(e - d, c - b, f - e):
                print(b + e, c + e)
            if mezi(b - a, f - e, c - b):
                print(b + e, b + f)
```

Tento algoritmus si můžeme ilustrovat na stejném příkladu jako v kapitole 2.4, tedy pro $P = [(0,1),(3,2),(6,0),(2,5)]$ a $Q = [(6,0),(9,2),(8,3),(9,4),(6,4),(4,7)]$. A dostaneme:

```
[6 1] [9 2]
[8 4] [11  5]
[6 5] [9 6]
[4 8] [6 1]
[9 2] [12  0]
[9 2] [12  4]
[12  4] [11  5]
[11  5] [14  3]
[11  5] [12  6]
[12  6] [9 6]
[9 6] [12  4]
[9 6] [7 9]
[7 9] [9 2]
[12  0] [15  2]
[15  2] [11  7]
[14  3] [15  4]
[15  4] [11  9]
[12  4] [10  7]
[10  7] [ 6 12]
[11  7] [10  8]
[10  8] [8 4]
[11  9] [8 9]
[8 9] [6 5]
[6 12] [4 8]
```

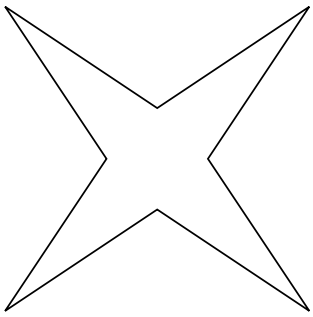
Vykreslíme-li si tyto orientované úsečky, tak dostaneme:



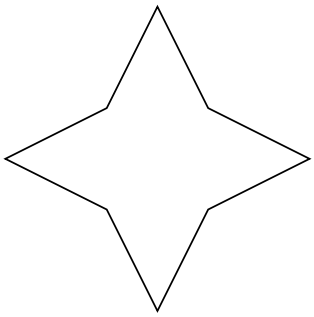

Obrázek 3.2: Konvoluce $P * Q$.

Díky (Wein, 2006, str. 3) víme následující. Konvoluce dvou konvexních mnohoúhelníků je jeden jednoduchý neprotínající se cyklus. Konvoluce jednoho konvexního a jednoho nekonvexního je opět jeden cyklus, který se ovšem může protínat.

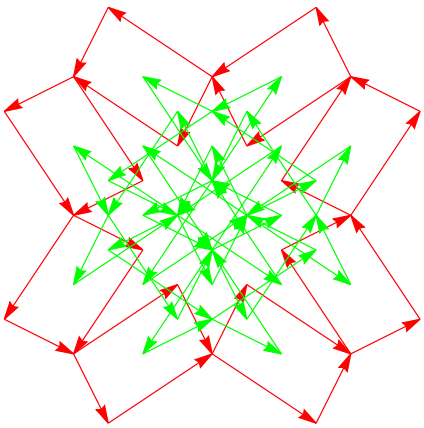
A konvoluce dvou nekonvexních mnohoúhelníků může být složena z více cyklů, jako například v tomto případě, kde máme $P = [(0,0), (3,2), (6,0), (4,3), (6,6), (3,4), (0,6), (2,3)]$ a $Q = [(0,3), (2,2), (3,0), (4,2), (6,3), (4,4), (3,6), (2,4)]$.



Obrázek 3.3: Mnohoúhelník $P$.



Obrázek 3.4: Mnohoúhelník $Q$.



Obrázek 3.5: Ukázka kdy konvoluce $P * Q$ obsahuje více cyklů.

## 3.2 Obtočnost

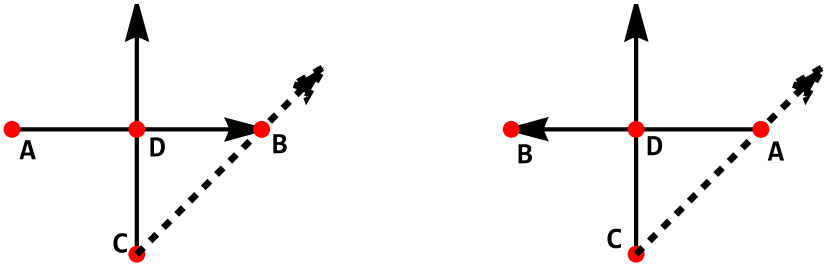
Mějme tedy nějakou konvoluci $P * Q$ sestávající z jednoho, či více konvolučních cyklů. Na tyto konvoluční cykly se můžeme dívat jako na uzavřené Eulerovské tahy a to tak, že krajní body a průsečíky orientovaných úseček z konvolučního cyklu ztotožníme s vrcholy grafu a orientované úsečky mezi těmito body ztotožníme s orientovanými hranami grafu. [4]

Minkowského součet dvou mnohoúhelníků je pak podle (Wein, 2006, str. 3) množina bodů, která má nenulovou obtočnost (winding number), vůči alespoň jednomu z těchto konvolučních cyklů. Tento pojem si nyní zadefinujeme. Budeme při tom vycházet z definice z (Grünbaum a Shephard, 1990, str. 178)

**Definice 17.** *Buď* $u = \overrightarrow{A, B}$ *orientovaná úsečka. Mějme bod* $C$*, který neleží na jedné přímce s body* $A, B$ *a buď* $D$ *bod ležící na úsečce* $u$*. Pak řekneme, že polopřímka* $R = [C, D)$ protíná $u$ po směru hodinových ručiček vzhledem k $C$,

[4] Pro účel této podkapitoly nazveme krajní body a průsečíky úseček z konvolučního cyklu vrcholy. Orientované úsečky mezi nimi pak nazveme hrany.

*pokud rotací $R$ po směru hodinových ručiček okolo bodu $C$ protneme dřív bod $B$, než bod $A$. V opačném případě řekneme, že $R$* protíná *$u$* v protisměru hodinových ručiček vzhledem k bodu $C$.




Obrázek 3.6: $R$ protíná $u$ po směru hodinových ručiček (vlevo), $R$ protíná $u$ v protisměru hodinových ručiček (vpravo).

**Definice 18.** *Buď $\boldsymbol{C}$ konvoluční cyklus a $A \in \mathbb{R}^2$ libovolný bod, který neleží na žádné hraně z $\boldsymbol{C}$. Mějme polopřímku $R = [A, B)$, která neobsahuje žádný vrchol z $\boldsymbol{C}$. Pro každou hranu $E_j$ definujeme* index $\Psi(R, E_j)$ *následovně:*

$$\Psi(R, E_j) = \begin{cases} 0 & R \text{ neprotíná } E_j \\ 1 & R \text{ protíná } E_j \text{ v protisměru hodinových ručiček} \\ -1 & R \text{ protíná } E_j \text{ po směru hodinových ručiček} \end{cases}$$

Obtočnost (winding number) *$w(\boldsymbol{C}, A)$ bodu $A$ vzhledem ke konvolučnímu cyklu $\boldsymbol{C}$ je pak rovna* $\sum_j \Psi(R, E_j)$

Na první pohled není vidět, že je tato definice korektní. Ke korektnosti potřebujeme, aby nám nezáleželo na zvoleném směru polopřímky R, tedy aby platilo $\sum_j \Psi(R_1, E_j) = \sum_j \Psi(R_2, E_j)$ pro libovolné polopřímky $R_1, R_2$ vycházející z bodu $A$ a neprocházející žádným vrcholem z $\mathbf{C}$.

*Důkaz.* Buď $A \in \mathbb{R}^2$ libovolný bod, který neleží na žádné hraně z $\mathbf{C}$, a $R = [A, B)$ polopřímka, která neobsahuje žádný vrchol z $\mathbf{C}$, a buď $W = \sum_j \Psi(R, E_j)$. Začneme rotovat polopřímku $R$ po směru hodinových ručiček okolo bodu $A$, čímž budeme dostávat nové polopřímky $R_i$. Takto rotujeme, dokud pro nějaké $k$, nebude $R_k$ protínat některý z vrcholů $\mathbf{C}$. Pro $R_k$ nemáme obtočnost zadefinovanou, nicméně si všimneme, že $W = \sum_j \Psi(R_i, E_j)$ pro všechny $i < k$. Kdyby tato rovnost neplatila, tak by některá $R_i$ musela protnout nějakou novou hranu oproti předchozím polopřímkám, ta by však musela vycházet z nějakého vrcholu $V$, který by nutně předchozí polopřímky musely protnout, to je však spor s tím, že $R_k$ je první taková, co protne nějaký vrchol.

Může se stát, že těch vrcholů bude na $R_k$ ležet více. Rozdělíme-li si nyní $W$ na dvě části, a to na součet indexů hran, které jdou do nebo z nějakého vrcholu na $R_k$, a součet indexů zbylých hran. Součet indexů zbylých hran se pro $R_{k+1}$ nijak nezmění (stejný argument jako výše).

Pro vrcholy z $R_k$, které mezi sebou nejsou spojeny hranou, se součet indexů hran, které jdou do nebo z těchto vrcholů také nezmění, protože i když se počet hran, které protne $R_{k+1}$ změní, tak díky tomu, že na konvoluční cyklus můžeme nahlížet jako na Eulerovský tah, tak na každou hranu, která jde do vrcholu připadá jedna, co jde z vrcholu, tedy jedna se započítá s indexem $+1$ a druhá s indexem $-1$, čímž se vyruší a obtočnost se nám nezmění.

Jsou-li dva vrcholy ležící na $R_k$ spojeny hranou, tak platí podobný argument jako výše, jen s tím rozdílem, že tato hrana se nezapočítá s žádným indexem. Tato hrana je však jednou výstupní hrana a jednou vstupní, tedy počet hran, které jdou do jednoho z těchto dvou vrcholů a počet hran, které vychází z jednoho z těchto dvou vrcholů je stejný a indexy se opět navzájem vynulují, tudíž se výsledné $W$ nezmění.

□

**Definice 19.** *Buď $\boldsymbol{C}$ konvoluční cyklus, pak komponenty souvislosti $\mathbb{R}^2 \setminus \boldsymbol{C}$ nazveme* oblasti.

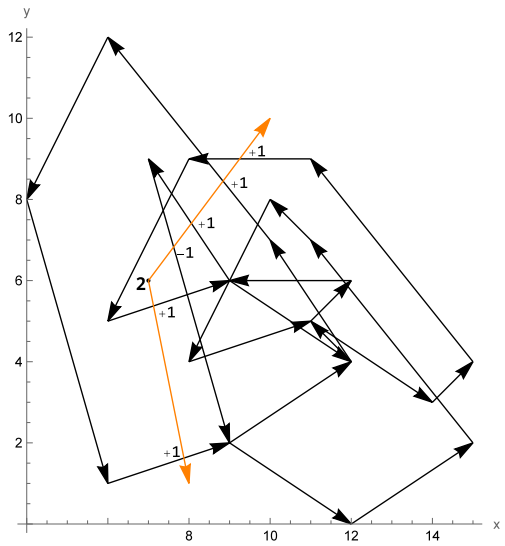
Z přednášek z matematické analýzy víme, že každé dva body ze stejné oblasti lze spojit lomenou čárou, která leží celá v dané oblasti.

**Lemma 20.** *Každé dva body v jedné oblasti mají stejnou obtočnost.*

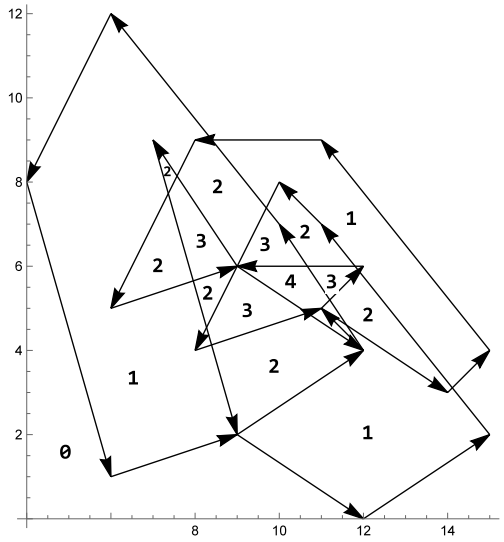
*Důkaz.* Mějme dva body $A$, $B$ v jedné oblasti, můžeme je tedy spojit lomenou čarou. Ukážeme, že každé dva po sobě jdoucí body mají stejnou obtočnost, tudíž i body $A$ a $B$ budou mít stejnou obtočnost. Nechť $A_1$ je prvním vrcholem na lomené čáře po bodu $A$. Vezmeme libovolný bod $C$, který leží na polopřímce $[A, A_1)$, ale ne mezi body $A$ a $A_1$. Pokud polopřímka $[A, A_1)$ neobsahuje žádný vrchol z konvolučního cyklu $\mathbf{C}$, tak polopřímky $[A, C)$ i $[A_1, C)$ protínají stejný počet hran ve stejném směru. To platí, protože všechny body na polopřímce $[A_1, C)$ jsou zároveň i na polopřímce $[A, C)$ a mezi body $A$ a $A_1$ není žádná hrana. V tomto případě tedy mají oba body $A$ i $A_1$ stejnou obtočnost. Pokud naopak polopřímka $[A, A_1)$ obsahuje nějaký vrchol z konvolučního cyklu $\mathbf{C}$, tak mezi body $A$ a $A_1$ nalezneme jinou lomenou čáru a postupujeme jako výše. Jelikož vrcholů v $\mathbf{C}$ je jen konečně mnoho a bodů v oblasti nekonečně mnoho, tak lze vždy nalézt lomenou čáru mezi body tak, aby při výše popsaném postupu vznikaly polopřímky neobsahující žádný vrchol z $\mathbf{C}$.

□

Tyto poznatky využijeme pro konvoluci 3.2, která nám vyšla v předchozí sekci.



Spočítáme-li nyní obtočnost pro každou z oblastí, dostaneme:




Podíváme-li se nyní na množinu bodů s nenulovou obtočností, dostaneme nepřekvapivě stejný výsledek, jako když jsme na tento příklad použili rozkladovou metodu 2.12.

# 4. Porovnání obou metod

Pro porovnání obou metod využijeme C++ knihovnu CGAL, ve které jsou implementovány oba algoritmy na Minkowského součet pro dva nekonvexní mnohoúhelníky. Konkrétně pak budeme pro rozkladovou metodu využívat tři rozkladové algoritmy Triangulaci, Small side angle bisector (budeme značit SSAB) a Greene convex decomposition (Greene). Na konvoluci pak použijeme Full Convolution (Konvoluce). Všechny tyto algoritmy jsou popsány v dokumentaci knihovny CGAL dostupné zde.

Dobu běhu výpočtu budeme měřit pomocí funkce,

```
auto start = high_resolution_clock::now();
auto stop = high_resolution_clock::now();
auto duration = duration_cast<milliseconds>(stop - start);
std::cout << duration.count() << std::endl;
```

která nám vrátí čas v milisekundách. Veškeré časové hodnoty v této kapitole budou tedy udávány v této jednotce. Výpočty budou probíhat na počítači Lenovo Yoga Slim 7i Pro X s 16 GB RAM a procesorem 12th Gen Intel(R) Core(TM) i5-12500H 2.50 GHz. Pro každý vstup spustíme algoritmus desetkrát a výslednou hodnotou v tabulce bude průměr (zaokrouhlený na celé číslo) těchto hodnot. Vstupy budou následující obrazce:

**Hrad**: Hrad je mnohoúhelník se 74 vrcholy, sestávající převážně ze svislých a vodorovných hran. Dále v textu ho budeme značit **Hr**.

**Rytíř**: Rytíř je tvořen 44 vrcholy a v textu ho budeme dále značit **R**.

**Had**: Mnohoúhelník ve tvaru hada obsahuje 37 vrcholů a v textu ho budeme značit **H**.

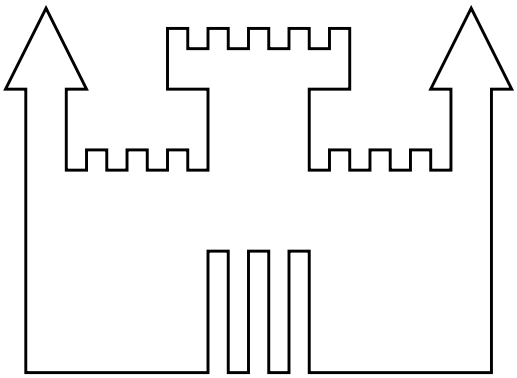
**Krakatice**: Krakatice je tvořena 97 vrcholy a budeme ji značit **K**.

**Jeskyně**: Jeskyni tvoří 57 vrcholů. Její součástí je několik zubů a úzkých skulin, které při součtu s některými ostatními mnohoúhelníky mohou vést ke zdlouhavějšímu výpočtu, např. při výpočtu děr. Budeme značit **J**.

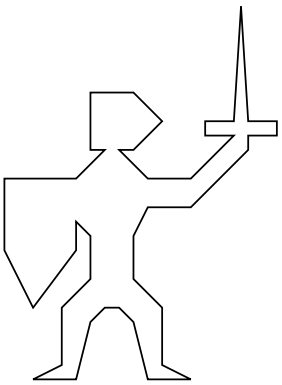
**Random Star-shaped**: Random Star-shaped mnohoúhelník si necháme vygenerovat pomocí kódu, který vezme vektor (1,0) a zrotuje ho o $\frac{2\pi}{n} + r$ stupňů proti směru hodinových ručiček, kde $r$ je náhodná hodnota z intervalu $(-\frac{2}{n}, \frac{2}{n})$. To uděláme $n - 1$ krát, čímž dostaneme $n$ vektorů, navíc ještě po každé rotaci přenásobíme předešlý vektor náhodným skalárem z intervalu $(0, n)$. Souřadnice takto vzniklých vektorů nám dají vrcholy mnohoúhelníku na $n$ vrcholech. Vygenerujeme si celkem tři takové s 10, 100 a 1000 vrcholy, které budeme postupně značit **Rnd10**, **Rnd100**, **Rnd1000**.

**Čtverec**. Čtverec, budeme značit **Č**, bude, jak již název napovídá, čtverec o délce strany 1. Ten využijeme jako referenční hodnotu při součtu s ostatními

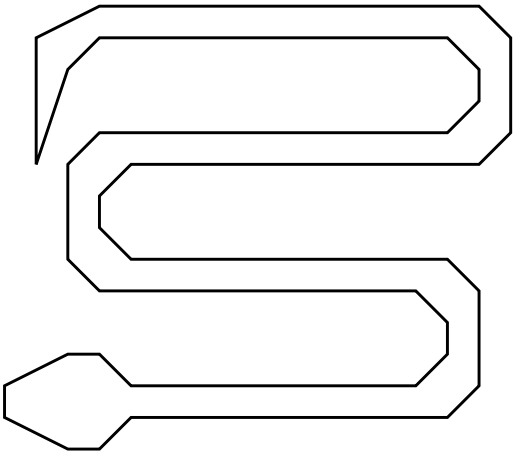
mnohoúhelníky.



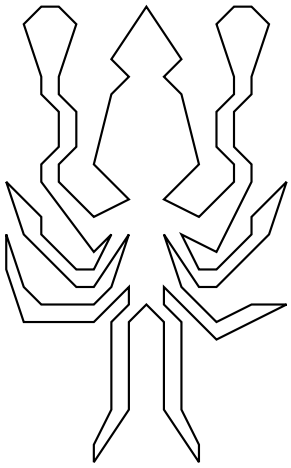
Obrázek 4.1: Hrad



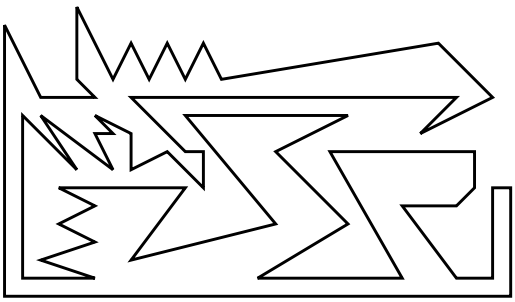
Obrázek 4.2: Rytíř



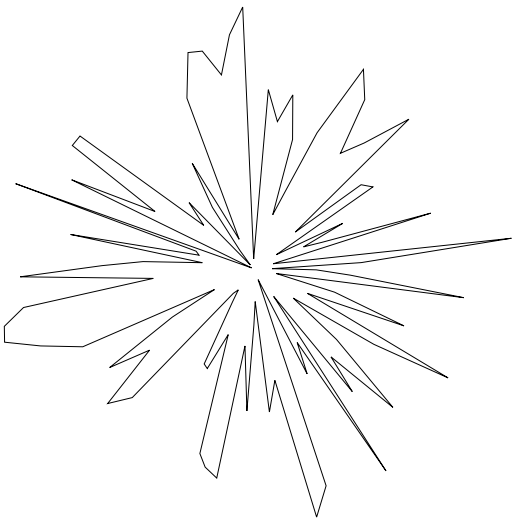
Obrázek 4.3: Had



Obrázek 4.4: Krakatice

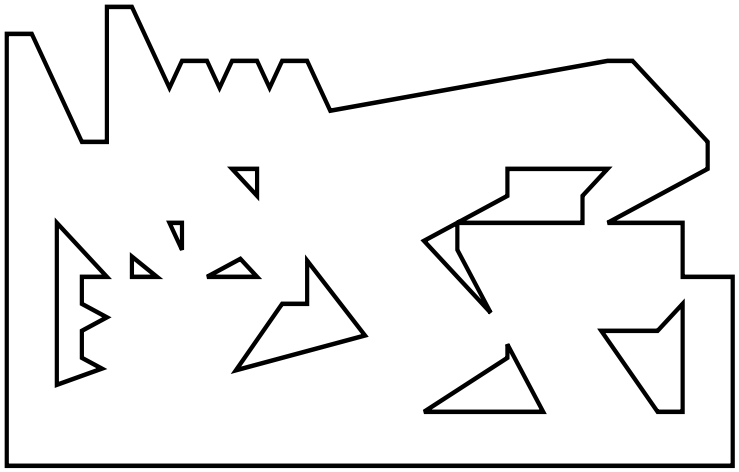


Obrázek 4.5: Jeskyně



Obrázek 4.6: Rnd100

Jak již bylo naznačeno výše, začneme tím, že každý z výše uvedených mnohoúhelníků sečteme se čtvercem, jakožto jedním z nejjednodušších mnohoúhelníků. Takto například dopadne součet **Č**⊕**J**. Tento příklad je o to zajímavější, že vzniklý mnohoúhelník obsahuje hned několik děr.



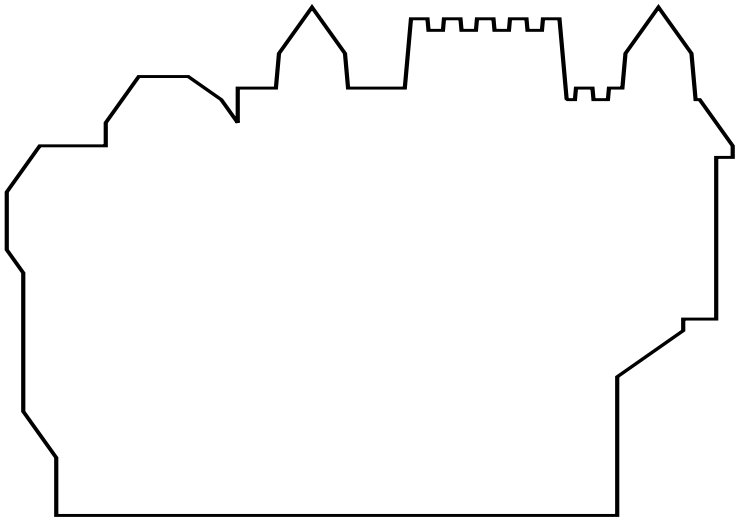
Obrázek 4.7: Součet čtverce a jeskyně.

| Mnohoúhelník ⊕ **Č** | Konvoluce | Triangulace | SSAB | Greene |
|---|---|---|---|---|
| **Hr** | **28** | 162 | 42 | 50 |
| **R** | **25** | 106 | 34 | 42 |
| **H** | **21** | 84 | 44 | Error |
| **K** | **63** | 279 | 113 | Error |
| **J** | **54** | 144 | 71 | 83 |
| **Rnd10** | **26** | 94 | 60 | 73 |
| **Rnd100** | **201** | 1 102 | 458 | 642 |
| **Rnd1000** | **2 103** | 24 687 | 62 708 | 14 116 |

Vidíme, že konvoluční metoda je pro součet nekonvexního mnohoúhelníku s jednoduchým konvexním násobně rychlejší než metody rozkladové. Nyní se podíváme, zda to tak bude platit i pro součet dvou složitějších nekonvexních mnohoúhelníků.

| Mnohoúhelníky | Konvoluce | Triangulace | SSAB | Greene |
|---|---|---|---|---|
| **R** ⊕ **Hr** | Error | 2 980 | **473** | 658 |
| **H** ⊕ **J** | Error | 1 971 | **677** | Error |
| **K** ⊕ **Hr** | 2 819 | 6 314 | **1 016** | Error |
| **K** ⊕ **Rnd100** | 168 021 | 177 671 | **37 650** | Error |
| **J** ⊕ **Rnd100** | 57 326 | 87 500 | **12 317** | 23 595 |
| **R** ⊕ **Rnd100** | **11 523** | 71 859 | 12 947 | 24 270 |
| **Rnd100** ⊕ **Rnd100** | 189 043 | 126 586 | **53 504** | 76 576 |
| **Rnd10** ⊕ **Rnd1000** | **4 069** | 111 028 | 51 919 | 69 732 |

I zde nám bohužel u pár vstupů vrátil CGAL Error. Ze zbytku si však můžeme všimnout, že při součtu dvou složitějších mnohoúhelníků již konvoluční metoda přestává být rychlejší a naopak je někdy i násobně pomalejší. Je-li však jeden



Obrázek 4.8: Součet rytíře a hradu.

ze vstupních mnohoúhelníků jednodušší a na méně vrcholech, pak je konvoluční metoda rychlejší, viz **Rnd10** $\oplus$ **Rnd1000**. Na určitý typ vstupů se tedy jedná o velmi efektivní metodu na výpočet Minkowského součtu dvou nekonvexních mnohoúhelníků. Pro úplnost uveďme, že časové odhady jsou popsány v Wein (2006). Rychlost běhu algoritmu velmi závisí hned na několika faktorech, z hlavních uveďme počty vrcholů vstupních mnohoúhelníků, velikost jejich konvoluce, počet nekonvexních úhlů vstupních mnohoúhelníků, či počtu konvexních mnohoúhelníků, na které se vstupní mnohoúhelníky rozloží.

# Závěr

Hlavním cílem této práce bylo popsat algoritmy pro Minkowského součet dvou nekonvexních mnohoúhelníků a porovnat je na vlastních vstupech pomocí C++ knihovny CGAL. K tomu patřilo i dostatečné vybudování teorie okolo tohoto matematického problému. I přes občasné mezery v teorii v citovaných článcích, ne zcela dostačující dokumentaci CGALu a občasné potíže při práci s touto knihovnou, se nám hlavní cíl práce podařilo splnit. Udělali jsme si tak dobrou představu o Minkowského součtech konvexních i nekonvexních mnohoúhelníků a zjistili jsme, že byť není konvoluční metoda ještě dobře popsána a dokázána v literatuře, tak na našich experimentálních vstupech fungovala pro určitý typ případů násobně rychleji než metoda rozkladová a má tedy ve výpočetní geometrii velký potenciál.

# Seznam použité literatury


Agarwal, P. K., Flato, E. a Halperin, D. (2002). Polygon decomposition for efficient construction of minkowski sums. *Computational Geometry*, **21** (1-2), 39–61.

Baram, A., Fogel, E., Halperin, D., Hemmer, M. a Morr, S. (2018). Exact minkowski sums of polygons with holes. *Computational Geometry*, **73**, 46–56.

de Berg, M., Cheong, O., van Kreveld, M. a Overmars, M. (2008). *Computational geometry algorithms and applications*. Spinger.

Dobkin, D., Hershberger, J., Kirkpatrick, D. a Suri, S. (1990). Implicitly searching convolutions and computing depth of collision. In *International Symposium on Algorithms*, pages 165–180. Springer.

Fradelizi, M., Madiman, M., Marsiglietti, A. a Zvavitch, A. (2018). The convexification effect of minkowski summation. *EMS Surveys in Mathematical Sciences*, **5**(1), 1–64.

Greene, D. H. (1983). The decomposition of polygons into convex parts. *Computational geometry*, **1**, 235–259.

Grünbaum, B. a Shephard, G. C. (1990). Rotation and winding numbers for planar polygons and curves. *Transactions of the american Mathematical Society*, **322**(1), 169–187.

Guibas, L., Ramshaw, L. a Stolfi, J. (1983). A kinetic framework for computational geometry. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pages 100–111. IEEE Computer Society.

Hales, T. (2007). The jordan curve theorem, formally and informally. *American Mathematical Monthly*, **114**. doi: 10.1080/00029890.2007.11920481.

Kaul, A. (1991). Computing minkowski sums of regular polygons. In *Proc. 3rd Canadian Conf. on Computational Geometry*, pages 74–77.

Kaul, A. (1992). Minkowski sums: A simulation tool for cad/cam. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 9358, pages 447–456. American Society of Mechanical Engineers.

Lee, I., Lee, K. K., Sim, O., Woo, K. S., Buyoun, C. a Oh, J.-H. (2015). Collision detection system for the practical use of the humanoid robot. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 972–976. IEEE.

Lengyel, J., Reichert, M., Donald, B. R. a Greenberg, D. P. (1990). Real-time robot motion planning using rasterizing computer graphics hardware. *ACM Siggraph Computer Graphics*, **24**(4), 327–335.

Matousek, J. (2013). *Lectures on discrete geometry*, volume 212. Springer Science & Business Media.

Wein, R. (2006). Exact and efficient construction of planar minkowski sums using the convolution method. In *Algorithms–ESA 2006: 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006. Proceedings 14*, pages 829–840. Springer.

# Seznam obrázků