VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MATEMATIKY
INSTITUTE OF MATHEMATICS

KINEMATIKA ROBOTICKÉHO RAMENE POMOCÍ GEOMETRICKÝCH ALGEBER

KINEMATICS OF A ROBOTIC ARM BY MEANS OF GEOMETRIC ALGEBRAS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MICHAL KŘÁPEK**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Mgr. PETR VAŠÍK, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav matematiky
Student:	**Michal Křápek**
Studijní program:	Matematické inženýrství
Studijní obor:	bez specializace
Vedoucí práce:	**doc. Mgr. Petr Vašík, Ph.D.**
Akademický rok:	2021/22

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Kinematika robotického ramene pomocí geometrických algeber

Stručná charakteristika problematiky úkolu:

Geometrické algebry jsou Cliffordovy algebry takové, že geometrické objekty eukleidovského prostoru jsou jejich prvky současně s eukleidovskými transformacemi. Tento přístup lze následně využít pro manipulaci s reálnými objekty, například roboty. Kinematika robotického ramene patří již k tradičnímu problému, kde se geometrické algebry používají.

Cíle bakalářské práce:

Získání základních znalostí aparátu geometrických algeber a jejich aplikace na vybraném problému, konkrétně bude zkoumána dopředná a inverzní kinematika robotického ramene.
Zpracování algoritmu v Pythonu nebo podobném programovacím jazyce s využitím geometrických algeber.

Seznam doporučené literatury:

DORST, L., FONTIJNE, D., MANN, S. Geometric Algebra for Computer Science: an Object-Oriented Approach to Geometry. Rev. ed. Burlington, Mass.: Morgan Kaufmann Publishers,c2007. Morgan Kaufmann series in computer graphics. ISBN 978-0-12-374942-0.

GONZÁLEZ CALVET, R. Treatise of Plane Geometry Through Geometric Algebra. 1. Cerdanyola del Vallés: [nakladatel není známý], 2007. TIMSAC. ISBN 978-84-611-9149-9.

HILDENBRAND, D. Foundations of Geometric Algebra Computing. Geometry and Computing, 8. ISBN 3642317936.

HILDENBRAND, D. Introduction to Geometric Algebra Computing. Boca Raton, 2018. ISBN 978-149-8748-384.

PERWASS, C. Geometric Algebra with Applications in Engineering. Berlin: Springer, c2009. ISBN 354089067X.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2021/22

V Brně, dne

L. S.

prof. RNDr. Josef Šlapal, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

V této práci se zabýváme dopřednou inverzní kinematiku robotického ramene za pomocí modelu dvoudimenzionálního prostoru v konformní geometrické algebře. Cílem této práce je návrh algoritmů k řešení tohoto problému a jejich následná implementace.

Výsledkem je pět algoritmů implementovaných v jazyce python, konkrétně jeden pro výpočet polohy ramene a čtyři pro výpočet trajektorie gripperu ramene.

V práci byl problém vyřešen za pomoci věty o orientaci normály a s pomocí signatury trojúhelníku, díky čemuž byla snížena výpočetní náročnost nejkomplexnějšího algoritmu, který kombinuje pohyb gripperu po lomené čáře s pohybem gripperu po kružnicích. Přínosem této práce je nový pohled na řešení inverzního kinematického problému.

Abstract

In this thesis we are dealing with forward and inverse kinematics of a robotic arm using a model of two dimensional space in conformal geometric algebra. Goal of this thesis is a proposal of algorithms for dealing with inverse kinematics problem and their implementaion.

Five algorithms were constructed and implemented in python language. One for computing a position of a robotic arm and four for calculating the trajectory of the gripper.

In this thesis, the problem was solved using a theorem about orientation of the line segment normal and with the triangle signature. Because of that, the cumputing load was reduced in implementation of the most complex algorithm which is combining the motion of the gripper along a polygonal chain and motion of the gripper along circular trajectory. The benefit of this thesis is a new approach to solving inverse kinematics problem.

Klíčová slova

Konformní geometrická algebra, inverzní kinematika, robotické rameno, python.

Keywords

Conformal geometric algebra, inverse kinematics, robotic arm, python.

Citace

KŘÁPEK, Michal. *Kinematika robotického ramene pomocí geometrických algeber*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství. Vedoucí práce doc. Mgr. Petr Vašík, Ph.D.

Kinematika robotického ramene pomocí geometrických algeber

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci *Kinematika robotického ramene pomocí geometrických algeber* vypracoval samostatně pod vedením pana doc. Mgr. Petra Vašíka, Ph.D. s použitím materiálů uvedených v seznamu literatury.

.....
Michal Křápek
20. května 2022

Poděkování

Chtěl bych poděkovat především doc. Mgr. Petru Vašíkovi, Ph.D. za cenné připomínky a rady při vypracování bakalářské práce a za všechny možnosti, které mně v rámci práce nad rámec umožnil. Také bych chtěl poděkovat své rodině za podporu a Petře Bajerové za jazykové korektury pravopisné a gramatické.

Obsah

Kapitola 1

Úvod

Robotická ramena nalézají mnoho uplatnění, ať už v oblasti automatizace výroby nebo dokonce i medicíny. Úspěšně nahrazují lidský faktor výroby, zefektivňují její rychlost a přesnost. Jsou v drtivé většině ovládány výpočetní technikou, na kterou kladou nemalé výpočetní nároky, a proto zlepšovat možnosti řízení těchto robotických ramen je přirozeným požadavkem.

Tato práce má přispět ke zlepšení těchto řídících procesů za pomoci pokročilé matematiky, přesněji geometrické algebry, a přispět k nahrazení aktuálních výpočetních způsobů pomocí matic. Z hlediska počítačové výpočetní náročnosti je totiž řešení s pomocí matic mnohem výpočetně náročnější, než řešení s pomocí geometrických algeber.

Této práci jsem se začal věnovat, protože pracuji ve výrobě jednoúčelových strojů k automatizaci výroby, kde robotická ramena velice často využíváme. Proto řešení inverzního kinematického problému, který se zabývá otázkou, jak pohybovat robotickým ramenem, tak aby vykonalo konkrétní činnost nebo pohyb, je mě velice blízkým tématem.

Cílem této práce je tedy prostudování geometrických algeber a jejich následné využití ke konstrukci algoritmů pro výpočet inverzní kinematiky robotického ramene a jejich implementace v podobě krátkého výpočetního kódu.

V rámci práce bude sestrojen algoritmus pro pohyby nástroje robotického ramene po kružnici a přímce, ale budou sestrojeny i komplexnější algoritmy pohybu po lomené čáře a algoritmus pohybu po lomené čáře, kde jsou ostré rohy pohybu nahrazeny pohybem po kružnicích, ke kterému jsem využil i vlastních teoretických poznatků.

V první kapitole rozeberu konstrukci geometrické algebry, její základní operace a principy. V druhé se budu zabývat již konkrétní geometrickou algebrou, potřebnou k sestrojení požadovaných algoritmů a vysvětlením potřebných pojmů pro výpočty v jazyce python. V poslední kapitole provedu návrh algoritmů k řešení problému pohybu robotického ramene.

Kapitola 2

Teoretický úvod do geometrických algeber

Geometrická algebra je Cliffordova algebra, kde s pomocí kvadratické formy a zobrazení bodu z $\mathbb{R}^n$ do této algebry modelujeme geometrii. V případě eukleidovské geometrie prvky této algebry reprezentují geometrické objekty a eukleidovské transformace těchto objektů jako rotace a translace.

2.1 Bilineární a kvadratické formy

K zavedení geometrické algebry bude třeba připomenout vlastnosti kvadratických a bilineárních forem.[1]

Definice 1. Buď $\mathbb{R}^n$ vektorový prostor. Zobrazení $\langle , \rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ se nazývá *bilineární forma*, pokud je lineární v obou složkách, tedy pokud

$$\langle \alpha\mathbf{x} + \beta\mathbf{y}, \mathbf{z}\rangle = \alpha\langle \mathbf{x}, \mathbf{z}\rangle + \beta\langle \mathbf{y}, \mathbf{z}\rangle \tag{2.1}$$

a

$$\langle \mathbf{z}, \alpha\mathbf{x} + \beta\mathbf{y}\rangle = \alpha\langle \mathbf{z}, \mathbf{x}\rangle + \beta\langle \mathbf{z}, \mathbf{y}\rangle \tag{2.2}$$

pro všechny $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ a $\alpha, \beta \in \mathbb{R}$. Dále bilineární formu nazveme:

1. *symetrickou*, jestliže
$$\langle \mathbf{x}, \mathbf{y}\rangle = \langle \mathbf{y}, \mathbf{x}\rangle$$
pro všechna $\mathbf{x}, \mathbf{y} \in V$,
2. *antisymetrickou*, jestliže
$$\langle \mathbf{x}, \mathbf{y}\rangle = -\langle \mathbf{y}, \mathbf{x}\rangle$$
pro všechna $\mathbf{x}, \mathbf{y} \in V$,
3. *alternující*, jestliže
$$\langle \mathbf{x}, \mathbf{x}\rangle = 0$$
pro všechna $\mathbf{x} \in V$.

Bilineární formě, která je buď symetrická, antisymetrická nebo alternující také říkáme *vnitřní součin*.

Vnitřní součin na vektorech se nazývá skalární součin, protože výsledkem této operace je skalár.

Každou bilineární formu lze dále jednoznačně zadat pomocí matice.

Věta 2. *Buď $\mathcal{B} = (\mathbf{e_1}, ..., \mathbf{e_n})$ uspořádaná báze vektorového prostoru $\mathbb{R}^n$, kde $\mathbf{e_i}$ značí bázový vektor $(0, ..., 1, ..., 0)$ s 1 na i-tém místě, s vnitřním součinem $\langle , \rangle$, pak bilineární forma zadávající vnitřní součin $\langle , \rangle$ je jednoznačně určena maticí $M_\mathcal{B}$ o rozměrech $n \times n$ s prvky*

$$M_\mathcal{B} = (a_{ij}) = (\langle \mathbf{e_i}, \mathbf{e_j} \rangle),$$

kde $i = 1, ..., n$ a $j = 1, ..., n$.

Pro ukázku uveďme příklad symetrické bilineární formy.

Příklad 1. Buď $\mathbb{R}^3$ vektorový prostor s vnitřním součinem $\langle , \rangle$, zadaným pomocí matice M, kde

$$M = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Vnitřní součin dvou vektorů $\mathbf{u}, \mathbf{v} \in \mathbb{R}^3$ lze pak spočítat

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u} M \mathbf{v}.$$

A vnitřní součiny bázových vektorů jsou

$$\begin{aligned} &\langle \mathbf{e_1}, \mathbf{e_1} \rangle = \langle \mathbf{e_2}, \mathbf{e_2} \rangle = \langle \mathbf{e_3}, \mathbf{e_3} \rangle = 1, \\ &\langle \mathbf{e_2}, \mathbf{e_1} \rangle = \langle \mathbf{e_1}, \mathbf{e_2} \rangle = 2, \\ &\langle \mathbf{e_3}, \mathbf{e_1} \rangle = \langle \mathbf{e_1}, \mathbf{e_3} \rangle = \langle \mathbf{e_2}, \mathbf{e_3} \rangle = \langle \mathbf{e_3}, \mathbf{e_2} \rangle = 0. \end{aligned}$$

Pomocí těchto vztahů, lze spočítat vnitřní součin dvou vektorů z $\mathbb{R}^3$, a to díky linearitě bilineární formy (2.1), (2.2), kdy vnitřní součin rozložíme na součet vnitřních součinů bázových vektorů, které již jsou maticí určené. Tedy pro náš případ lze ukázat, že vnitřní součin dvou vektorů $\alpha\mathbf{e_1} + \beta\mathbf{e_2} + \gamma\mathbf{e_3}$ a $a\mathbf{e_1} + b\mathbf{e_2} + c\mathbf{e_3}$ je

$$\begin{aligned} \langle \alpha\mathbf{e_1} + \beta\mathbf{e_2} + \gamma\mathbf{e_3}, a\mathbf{e_1} + b\mathbf{e_2} + c\mathbf{e_3} \rangle &= \alpha a \langle \mathbf{e_1}, \mathbf{e_1} \rangle + \alpha b \langle \mathbf{e_1}, \mathbf{e_2} \rangle + \alpha c \langle \mathbf{e_1}, \mathbf{e_3} \rangle + \\ &+ \beta a \langle \mathbf{e_2}, \mathbf{e_1} \rangle + \beta b \langle \mathbf{e_2}, \mathbf{e_2} \rangle + \beta c \langle \mathbf{e_2}, \mathbf{e_3} \rangle + \\ &+ \gamma a \langle \mathbf{e_3}, \mathbf{e_1} \rangle + \gamma b \langle \mathbf{e_2}, \mathbf{e_2} \rangle + \gamma c \langle \mathbf{e_3}, \mathbf{e_3} \rangle \\ &= \alpha a + 2\alpha b + 2\beta a + \beta b + \gamma c, \end{aligned}$$

kde $\alpha, \beta, \gamma, a, b, c \in \mathbb{R}$.

Definice 3. *Kvadratická forma* na vektorovém prostoru $\mathbb{R}^n$ je bilineární zobrazení $Q : \mathbb{R}^n \to \mathbb{R}$ s následujícími vlastnostmi:

1. Pro všechna $\alpha \in \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^n$,
$$Q(\alpha\mathbf{x}) = \alpha^2 Q(\mathbf{x}).$$
2. Zobrazení
$$\langle \mathbf{x}, \mathbf{y} \rangle_Q = Q(\mathbf{x} + \mathbf{y}) - Q(\mathbf{x}) - Q(\mathbf{y})$$
je symetrická bilineární forma, která se nazývá *asociovaná* ke kvadratické formě Q.

Každá kvadratická forma lze reprezentovat symetrickou maticí, protože se jedná o speciální případ bilineární formy. Nyní uveďme ukázku nalezení asociované bilineární formy.

Příklad 2. Nechť máme dva obecné vektory o souřadnicích $(\alpha, \beta, \gamma), (a, b, c) \in \mathbb{R}^3$ a kvadratickou formu $Q(\mathbf{x}) = 5x_1^2 + 3x_2^2 + 4x_3^2 + 2x_1x_2 + 2x_2x_3$ pro $\mathbf{x} = (x_1, x_2, x_3) \in \mathbb{R}^3$. Pak Q lze reprezentovat maticí

$$Q(\mathbf{x}) = \begin{pmatrix} 5 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 4 \end{pmatrix}.$$

Potom příslušnou asociovanou bilineární formu získáme následovně:

$$\begin{aligned}
\langle(\alpha, \beta, \gamma), (a, b, c)\rangle_Q &= Q((\alpha, \beta, \gamma) + (a, b, c)) - Q((\alpha, \beta, \gamma)) - Q((a, b, c)) = \\
&= 5(\alpha + a)^2 + 3(\beta + b)^2 + 4(\gamma + c)^2 + 2(\alpha + a)(\beta + b) + \\
&\quad + 2(\beta + b)(\gamma c) - 5\alpha^2 - 3\beta^2 - 4\gamma - 2\alpha\beta - 2\beta\gamma - 5a^2 - \\
&\quad - 3b^2 - 4c^2 - 2ab - 2bc = \\
&= 5\alpha^2 + 10\alpha a + 5a^2 + 3\beta^2 + 6\beta b + 3b^2 + 4\gamma^2 + 8\gamma c + \\
&\quad + 4c^2 + 2\alpha\beta + 2\alpha b + 2a\beta + 2ab + 2\beta\gamma + 2\beta c + 2b\gamma + 2bc - \\
&\quad - 5\alpha^2 - 3\beta^2 - 4\gamma - 2\alpha\beta - 2\beta\gamma - 5a^2 - 3b^2 - 4c^2 - 2ab - 2bc = \\
&= 10\alpha a + 6\beta b + 8\gamma c + 2\alpha b + 2a\beta + 2\beta c + 2b\gamma.
\end{aligned}$$

Z koncového výrazu lze vidět maticový tvar asociované bilineární formy:

$$\langle \mathbf{x}, \mathbf{y}\rangle_Q = \begin{pmatrix} 10 & 2 & 0 \\ 2 & 6 & 2 \\ 0 & 2 & 8 \end{pmatrix}.$$

Věta 4. *[5] Nechť Q je symetrická bilineární forma (striktněji kvadratická forma), na vektorovém prostoru $\mathbb{R}^n$. Potom existuje báze $\mathbb{R}^n$ a nezáporná čísla p a q, tak že $p + q = r$, kde r je řád matice Q tak, že je-li Q reprezentováno maticí $B = (b_{ij})$, pak*

$$\begin{aligned}
b_{ii} &= 1 \text{ pro } 1 \leq i \leq p, \\
b_{ii} &= -1 \text{ pro } p + 1 \leq i \leq p + q, \\
b_{ij} &= 0 \text{ v ostatních případech.}
\end{aligned}$$

Čísla $p, q, (r - p - q)$ potom nazýváme signaturou bilineární, případně kvadratické, formy.

Z definice tedy plyne že každá kvadratická forma Q na $\mathbb{R}^n$ definuje symetrickou bilineární formu $\langle \mathbf{x}, \mathbf{y}\rangle_Q$. Této vlastnosti později využijeme při konstrukci vnitřního součinu na geometrické algebře. Dále navíc platí

Věta 5. *Ke každé bilineární formě $\langle , \rangle$ na vektorovém prostoru $\mathbb{R}^n$ existuje právě jedna symetrická a antisymetrická bilineární forma $\langle , \rangle_S$ a $\langle , \rangle_A$ tak, že $\langle , \rangle = \langle , \rangle_S + \langle , \rangle_A$ a navíc*

$$\begin{aligned}
\langle \mathbf{x}, \mathbf{y}\rangle_S &= \frac{1}{2}(\langle \mathbf{x}, \mathbf{y}\rangle + \langle \mathbf{y}, \mathbf{x}\rangle), \\
\langle \mathbf{x}, \mathbf{y}\rangle_A &= \frac{1}{2}(\langle \mathbf{x}, \mathbf{y}\rangle - \langle \mathbf{y}, \mathbf{x}\rangle)
\end{aligned}$$

pro všechna $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.

Nyní uveďme příklad tohoto rozkladu na symetrickou a antisymetrickou část. Volili jsme jinou matici M určující bilineární formu, jelikož matice M z předchozího příkladu byla symetrická a z toho tedy triviálně vyplývá, že antisymetrickou bilineární formou by byla nulová matice.

Příklad 3. Nechť $\langle , \rangle$ je bilineární forma daná maticí M na vektorovém prostoru $\mathbb{R}^3$ nad polem $\mathbb{R}$, kde

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 2 \\ 4 & 0 & 3 \end{pmatrix}.$$

Potom symetrickou část podle Věty 5 vypočteme následovně:

$$\langle , \rangle_S = \frac{1}{2}\left(\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 2 \\ 4 & 0 & 3 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 4 \\ 0 & 2 & 0 \\ 0 & 2 & 3 \end{pmatrix}\right) = \frac{1}{2}\begin{pmatrix} 2 & 0 & 4 \\ 0 & 4 & 2 \\ 4 & 2 & 6 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 2 & 1 \\ 2 & 1 & 3 \end{pmatrix}$$

a antisymetrickou část

$$\langle , \rangle_A = \frac{1}{2}\left(\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 2 \\ 4 & 0 & 3 \end{pmatrix} - \begin{pmatrix} 1 & 0 & 4 \\ 0 & 2 & 0 \\ 0 & 2 & 3 \end{pmatrix}\right) = \frac{1}{2}\begin{pmatrix} 0 & 0 & -4 \\ 0 & 0 & 2 \\ 4 & -2 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & -2 \\ 0 & 0 & 1 \\ 2 & -1 & 0 \end{pmatrix}.$$

Správnost ověříme tak, že jejich součet dává původní matici M, tedy

$$\langle , \rangle = \left(\begin{pmatrix} 1 & 0 & 2 \\ 0 & 2 & 1 \\ 2 & 1 & 3 \end{pmatrix} + \begin{pmatrix} 0 & 0 & -2 \\ 0 & 0 & 1 \\ 2 & -1 & 0 \end{pmatrix}\right) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 2 \\ 4 & 0 & 3 \end{pmatrix}.$$

Získali jsme tedy rozklad bilineární formy na symetrickou a antisymetrickou část.

2.2 Geometrická algebra

Zaměříme se pouze na geometrické algebry nad reálnými čísly s konečnými dimenzemi.

Definice 6 (Geometrická algebra)**.** *Geometrická algebra* (také zvaná *Cliffordova algebra*) je volná unitární asociativní algebra generovaná vektorovým prostorem $\mathbb{R}^n$ s kvadratickou formou Q a součinem $\circ$, pro který platí

$$\mathbf{u} \circ \mathbf{u} = \langle \mathbf{u}, \mathbf{u} \rangle_Q,$$

kde $\mathbf{u} \in \mathbb{R}^n$.

Geometrickou algebru budeme značit symbolem $\mathbb{G}_{p,q}$, kde p, q je signatura kvadratické formy na geometrické algebře dle Věty 4. Lze v indexu uvádět i třetí číslo ze signatury, nicméně v této práci se budeme zabývat pouze algebrami, kde se toto číslo rovná nule a obecně rovná-li se nule, tak se tento index i v ostatních publikacích neuvádí. Lze ukázat, že převodem na matici určující signaturu získáme ortonormální bázi vektorového prostoru generujícího $\mathbb{G}_{p,q}$, ve formě řádků.

Nyní identifikujeme operace na této algebře. Začneme vnějším součinem.

2.2.1 Vnější součin

Vnější součin je operace, která bude pro vektory a později i pseudoskalár reprezentovat objemy, a řekne nám mnohé o struktuře geometrické algebry. Budeme čerpat především z [11]. Nejdříve tyto operace zavedeme pro vektory a následně je nadefinujeme pro celou geometrickou algebru.

Definice 7 (Vnější součin)**.** Nechť $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ a $\beta \in \mathbb{R}$. Potom zobrazení $\wedge : \mathbb{R}^n \wedge \mathbb{R}^n \to \bigwedge^2 \mathbb{R}^n$, s vlastnostmi

$$\begin{aligned}&\text{Antisymetrie: } \mathbf{u} \wedge \mathbf{v} = -\mathbf{u} \wedge \mathbf{v}\\ &\text{Škálování: } \mathbf{u} \wedge (\beta\mathbf{v}) = \beta(\mathbf{u} \wedge \mathbf{v})\\ &\text{Distributivita: } \mathbf{u} \wedge (\mathbf{v} + \mathbf{w}) = (\mathbf{u} \wedge \mathbf{v}) + (\mathbf{u} \wedge \mathbf{w})\end{aligned}$$

kde $\bigwedge^2 \mathbb{R}^n$ je lineární prostor dimenze $\binom{n}{2}$, nazýváme *vnější součin*. Operátor $\wedge$ se čte anglicky "*wedge*".
Vnější součin lze zavést také jako zobrazení $\wedge : \mathbb{R}^n \wedge \mathbb{R}^n \wedge \mathbb{R}^n \to \bigwedge^3 \mathbb{R}^n$, pro které navíc platí

$$\text{Asociativita: } (\mathbf{u} \wedge \mathbf{v}) \wedge \mathbf{w} = \mathbf{u} \wedge (\mathbf{v} \wedge \mathbf{w})$$

kde $\bigwedge^3 \mathbb{R}^n$ je lineární prostor dimenze $\binom{n}{2}$.

Vnější součin tří vektorů z $\mathbb{R}^n$ můžeme chápat jako orientovaný objem, respektive dvou vektorů z $\mathbb{R}^2$, jako orientovanou plochu . Poznamenejme, že vnější součin se někdy zavádí bez asociativity. Z antisymetrie vyplývá, že vnější součin dvou stejných vektorů, respektive lineárně závislých vektorů, je nula, což je jedna z požadovaných geometrických vlastností. Vnější součin dvou vektorů nazveme bivektor. Analogicky tří vektorů trivektor, k vektorů k-vektor apod. Tyto prvky můžeme také nazývat anglicky blady stupně k, kde k je počet vektorů, mezi kterými tento vnější součin provádíme. Díky linearitě vnějšího součinu pak můžeme definovat prostor vnějších součinů.

Definice 8 (Prostor vnějších součinů k vektorů)**.** Buď $\mathbb{R}^n$ vektorový prostor a vnější součin zobrazení z kartézského součinu k vektorových prostorů do prostoru k-vektorů značeného $\bigwedge^k \mathbb{R}^n$ tj. $\wedge : \mathbb{R}^n \times \mathbb{R}^n \times ... \times \mathbb{R}^n \to \bigwedge^k \mathbb{R}^n$. Pak prostor $\bigwedge^k \mathbb{R}^n$ nazýváme prostor vnějších součinů k-vektorů, nebo ho také nazýváme např. bivektorový prostor pro $k = 2$, trivektorový prostor pro $k = 3$ apod.

Ukažme nyní zajímavou vlastnost vnějšího součinu.

Příklad 4. Buď $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2$, s bází $(\mathbf{e_1}, \mathbf{e_2})$ prostoru $\mathbb{R}^2$, tedy $\mathbf{u} = u_1\mathbf{e_1} + u_2\mathbf{e_2}$ a $\mathbf{v} = v_1\mathbf{e_1} + v_2\mathbf{e_2}$, kde $u_1, u_1, v_1, v_2 \in \mathbb{R}$. Pak vnější součin spočítáme jako

$$\begin{aligned}\mathbf{u} \wedge \mathbf{v} &= (u_1\mathbf{e_1} + u_2\mathbf{e_2}) \wedge (v_1\mathbf{e_1} + v_2\mathbf{e_2})\\ &= u_1v_1\mathbf{e_1} \wedge \mathbf{e_1} + v_1u_2\mathbf{e_1} \wedge \mathbf{e_2} + v_2v_1\mathbf{e_2} \wedge \mathbf{e_1} + u_2v_2\mathbf{e_2} \wedge \mathbf{e_2}\\ &= (u_1v_2 - u_2v_1)\mathbf{e_1} \wedge \mathbf{e_2}.\end{aligned}$$

Z výsledku lze vidět, že koeficient u výsledného wedge bázových vektorů je determinant matice skládající se z vektorů $\mathbf{u}, \mathbf{v}$ dimenze 2. Tedy můžeme vidět, že $\mathbf{u} \wedge \mathbf{v}$ má význam vážené, orientované oblasti dvoudimenzionálního prostoru.

Lze ukázat, že $\bigwedge^0 \mathbb{R}^n$ je 0-dimenzionální prostor skalárů, jelikož zůstane pouze škálovací

vlastnost z definice vnějšího součinu. Lze tedy říci, že skaláry jsou blady stupně 0. Podobně lze ukázat, že $\bigwedge^1 \mathbb{R}^n$ je vektorový prostor, ze kterého jsme $\bigwedge^k \mathbb{R}^n$ generovali. Poznamenejme, že generování gradovaného vektorového prostoru se zaručeně zastaví díky antikomutativitě a asociativitě vnějšího součinu v případě konečné dimenze vektorového prostoru, nad kterým vzniká. Demonstrujme to na příkladu.

Příklad 5. Nechť $\mathbb{R}^3$ je vektorový prostor s bází $(\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3})$. Potom provedeme-li vnější součin trivektoru a dalšího vektoru, tak už nemůžeme dostat 4-vektor, respektive vektor vyššího řádu, než je dimenze vektorového prostoru $\mathbb{R}^3$. Tedy pro $\alpha, \beta, \gamma \in \mathbb{R}$

$$\mathbf{e_1} \wedge \mathbf{e_2} \wedge \mathbf{e_3} \wedge (\alpha \mathbf{e_1} + \beta \mathbf{e_2} + \gamma \mathbf{e_3}) = \mathbf{e_1} \wedge \mathbf{e_2} \wedge \mathbf{e_3} \wedge \alpha \mathbf{e_1} + \mathbf{e_1} \wedge \mathbf{e_2} \wedge \mathbf{e_3} \wedge \beta \mathbf{e_2} + \mathbf{e_1} \wedge \mathbf{e_2} \wedge \mathbf{e_3} \wedge \gamma \mathbf{e_3} = 0.$$

Pro vektorový prostor tedy platí $\bigwedge \mathbb{R}^n = \bigoplus_k \bigwedge^k \mathbb{R}^n$ kde $k \in \{0, 1, ..., n\}$, a proto ho nazýváme gradovaný. Uveďme příklad uspořádané báze tohoto prostoru pro $n = 3$. Tato uspořádaná báze je

$$(1,\ \mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3},\ \mathbf{e_1} \wedge \mathbf{e_2}, \mathbf{e_2} \wedge \mathbf{e_3}, \mathbf{e_3} \wedge \mathbf{e_1},\ \mathbf{e_1} \wedge \mathbf{e_2} \wedge \mathbf{e_3}).$$

Jedná se tedy o osmidimenzionální prostor. Obecně lze říci, že pokud tento prostor generujeme z $\mathbb{R}^n$, pak dimenze bude 2^n.

Definice 9 (Pseudoskalár)**.** Blade nejvyššího stupně v bázi $\bigwedge R^n$ se nazývá *pseudoskalár*.

2.2.2 Geometrický součin

Definice 10 (Geometrický součin)**.** [13] Nechť $\mathbb{G}_{p,q}$ je geometrická algebra s kvadratickou formou Q. Operace této algebry se nazývá *geometrický součin*, neznačí se a definuje se jako $\mathbf{e_i e_i} = \langle \mathbf{e_i}, \mathbf{e_i} \rangle_Q$, kde $\mathbf{e_i} \in \mathbb{R}^{p+q}$ je bázový vektor.

Geometrický součin má pro vektory následující vlastnosti, které později rozšíříme na celý gradovaný prostor, respektive blady a skaláry.

Axiom 1. *[13] Nechť $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^n \subset \mathbb{G}_{p,q}$ a vnitřní součin $\cdot$ na $\mathbb{R}^n \subset \mathbb{G}_{p,q}$. Dále $\alpha \in \mathbb{R}$, pak:*

1. ***Uzavřenost algebry** vůči geometrickému součinu:*
$$\mathbf{uv} \in \mathbb{G}_{p,q}.$$
2. ***Asociativita**:*
$$(\mathbf{uv})\mathbf{w} = \mathbf{u}(\mathbf{vw}).$$
3. ***Distributivita**:*
$$\mathbf{u}(\mathbf{v} + \mathbf{w}) = \mathbf{uv} + \mathbf{uw} \ a \ (\mathbf{v} + \mathbf{w})\mathbf{u} = \mathbf{vu} + \mathbf{wu}.$$
4. ***Násobení skalárem**:*
$$\alpha \mathbf{u} = \mathbf{u} \alpha.$$
5. ***Definiční rovnice**:*
$$\mathbf{uu} = \mathbf{u} \cdot \mathbf{u}.$$

Lze nalézt v [13, 11], že tento součin má komutativní a antikomutativní část. Tuto vlastnost navíc používáme při generování geometrické algebry a k ukázce, že geometrická algebra má strukturu gradovaného prostoru. Zkusíme tedy pro vektory nyní najít tuto symetrickou a antisymetrickou část [13].

Buď $\mathbf{u}, \mathbf{v} \in \mathbb{G}_{p,q}$ jednavektory. Potom z bodu 3 a 5 Axiomu 1 a vlastností vnitřního součinu můžeme odvodit:

$$\begin{aligned} & (\mathbf{u}+\mathbf{v})(\mathbf{u}+\mathbf{v}) = (\mathbf{u}+\mathbf{v})\cdot(\mathbf{u}+\mathbf{v}) \\ \iff & \mathbf{uu}+\mathbf{uv}+\mathbf{vu}+\mathbf{vv} = \mathbf{u}\cdot\mathbf{u}+2\mathbf{u}\cdot\mathbf{v}+\mathbf{v}\cdot\mathbf{v} \\ \iff & \frac{1}{2}(\mathbf{uv}+\mathbf{vu}) = \mathbf{u}\cdot\mathbf{v} \end{aligned}$$

Výrazu $\frac{1}{2}(\mathbf{uv}+\mathbf{vu})$ se říká *antikomutátor* a budeme jej značit $\overline{\times}$. Podobnou konstrukcí se tvoří operand *komutátor* $\frac{1}{2}(\mathbf{uv}-\mathbf{vu})$, který budeme značit $\underline{\times}$. Součet těchto operací je geometrický součin vektorů.

$$\mathbf{uv} = \mathbf{u}\overline{\times}\mathbf{v} + \mathbf{u}\underline{\times}\mathbf{v} = \frac{1}{2}(\mathbf{uv}+\mathbf{vu}) + \frac{1}{2}(\mathbf{uv}-\mathbf{vu})$$

Jsou-li dva vektory ortogonální, tedy $\mathbf{u}\cdot\mathbf{v} = 0$, a vyjádříme-li to pomocí geometrického součinu, tak $\mathbf{u}\overline{\times}\mathbf{v} = 0$. Pro bázové vektory $\mathbf{e_i}, \mathbf{e_j} \in \mathbb{R}^n \subset \mathbb{G}_{p,q}$ dále platí:

$$\mathbf{e_i}\overline{\times}\mathbf{e_i} \neq 0 \text{ a } \mathbf{e_i}\overline{\times}\mathbf{e_j} = 0,\ i \neq j.$$

A dále pro $i \neq j$:

$$\begin{aligned} \mathbf{e_i e_j} &= \mathbf{e_i}\overline{\times}\mathbf{e_j} + \mathbf{e_i}\underline{\times}\mathbf{e_j} = \mathbf{e_i}\underline{\times}\mathbf{e_j} \\ \mathbf{e_j e_i} &= \mathbf{e_j}\overline{\times}\mathbf{e_i} + \mathbf{e_j}\underline{\times}\mathbf{e_i} = \mathbf{e_j}\underline{\times}\mathbf{e_i} \end{aligned}$$

a protože z definice $\overline{\times}$ plyne $\mathbf{u}\overline{\times}\mathbf{v} = -\mathbf{v}\overline{\times}\mathbf{u}$, potom

$$\mathbf{e_i e_j} = -\mathbf{e_j e_i}.$$

Prvek $\mathbf{e_i}\underline{\times}\mathbf{e_j}$ i $\mathbf{e_j}\underline{\times}\mathbf{e_i}$ není prvkem z $\mathbb{R}^n$. Geometrický součin nám tedy vygeneroval nový prvek, který patří do $\mathbb{G}_{p,q}$, což je jeden z prvních kroků k odhalení struktury geometrické algebry a toho, jak se generuje. Nyní uveďme příklad pro dva vektory $\mathbf{u} = u_1\mathbf{e_1} + u_2\mathbf{e_2}$ a $\mathbf{v} = v_1\mathbf{e_1} + v_2\mathbf{e_2}$, kde $\{\mathbf{e_1}, \mathbf{e_2}\} \subset \mathbb{R}^n$.

$$\begin{aligned} \mathbf{uv} &= (u_1\mathbf{e_1}+u_2\mathbf{e_2})(v_1\mathbf{e_1}+v_2\mathbf{e_2}) = (u_1v_1\mathbf{e_1e_1}+u_2v_2\mathbf{e_2e_2}) + (u_1v_2\mathbf{e_1e_2}+u_2v_1\mathbf{e_2e_1}) \\ &= (u_1v_1+u_2v_2) + (a_1v_2-u_2v_1)\mathbf{e_1e_2vu} \\ &= (v_1\mathbf{e_1}+v_2\mathbf{e_2})(u_1\mathbf{e_1}+u_2\mathbf{e_2}) \\ &= (v_1u_1\mathbf{e_1e_1}+v_2u_2\mathbf{e_2e_2}) + (v_1u_2\mathbf{e_1e_2}+v_2u_1\mathbf{e_2e_1}) \\ &= (u_1v_1+u_2v_2) - (a_1v_2-u_2v_1)\mathbf{e_1e_2}. \end{aligned}$$

Z těchto dvou výpočtů můžeme vidět, že

$$\mathbf{u}\overline{\times}\mathbf{v} = \frac{1}{2}(uv+vu) = u_1v_1+u_2v_2,$$

$$\mathbf{u}\underline{\times}\mathbf{v} = \frac{1}{2}(\mathbf{uv}-\mathbf{vu}) = (u_1v_2-u_2v_1)\mathbf{e_1e_2}.$$

Z těchto dvou rovnic lze vidět, že symetrickou částí geometrického součinu vektorů bude vnitřní součin a antisymetrickou bude vnější součin. Tedy $\mathbf{uv} = \mathbf{u}\cdot\mathbf{v} + \mathbf{u}\wedge\mathbf{v}$. Nicméně

tyto vlastnosti jsme zatím odvodili pouze pro vektory a bude třeba je rozšířit na multivektory. Zároveň díky vlastnostem vnějšího součinu víme a [13, 11], že geometrický součin vygeneruje gradovaný vektorový prostor a ten tedy bude strukturou geometrické algebry. Stupeň bladu $\mathbf{A} \in \mathbb{G}_{p,q}$ budeme značit jako $gr(\mathbf{A})$, podle anglického slova grade.

Definice 11 (Bázový blade)**.** Bázový blade $\mathbb{G}_{p,q}$, je geometrický součin rozdílných dvou a nebo více bázových vektorů $\mathbb{R}^{p+q}$.

Definice 12 (Projekce na stupeň)**.** Nechť $\mathbf{E}$ je bázový blade $\mathbb{G}_{p,q}$. Potom projekce bladu $\mathbf{E}$ na stupeň k se značí $\langle \mathbf{E} \rangle_k$ a definuje se

$$\langle \mathbf{E} \rangle_k = \begin{cases} \mathbf{E}, & gr(\mathbf{E}) = k \\ 0, & gr(\mathbf{E}) \neq k \end{cases}$$

přičemž tento operátor je distributivní a pro všechna $a \in \mathbb{R}$ platí

$$\langle a\mathbf{E} \rangle = a\langle \mathbf{E} \rangle. \tag{2.3}$$

Definice 13 (Vnitřní součin geometrické algebry)**.** Nechť $\mathbf{E_1}, \mathbf{E_2}$ jsou bázové blady $\mathbb{G}_{p,q}$ a nechť $gr(\mathbf{E_1}) = k$, $gr(\mathbf{E_2}) = l$. Potom *vnitřní součin* dvou bázových bladů se definuje jako

$$\mathbf{E_1} \cdot \mathbf{E_2} := \langle \mathbf{E_1 E_2} \rangle_{|k-l|} \ , \ k, l > 0,$$

a pro multivektory $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{G}_{p,q}$ má následující vlastnosti

$$\begin{aligned} \mathbf{A} \cdot (\mathbf{B} + \mathbf{C}) &= \mathbf{A} \cdot \mathbf{B} + \mathbf{A} \cdot \mathbf{C}, \\ (a\mathbf{A}) \cdot (b\mathbf{B}) &= ab(\mathbf{A} \cdot \mathbf{B}), \end{aligned}$$

kde $a, b \in \mathbb{R}$.

Tedy vnitřní součin dvou multivektorů $\mathbf{A}, \mathbf{B} \in \mathbb{G}_{p,q}$, kde $\mathbf{E_k}$ jsou bázové blady, $a_k, b_k \in \mathbb{R}$, $\mathbf{A} = \sum_k a_k \mathbf{E}_k$ a $\mathbf{B} = \sum_k b_k \mathbf{E}_k$, lze spočítat

$$\mathbf{A} \cdot \mathbf{B} = \sum_{k,l} a_k b_l (\mathbf{E}_k \cdot \mathbf{E}_l).$$

Uveďme příklad výpočtu.

Příklad 6. Nechť $\mathbf{B_1}, \mathbf{B_2} \in \mathbb{G}_{2,0}$, například $\mathbf{B_1} = 2\mathbf{e_1 e_2} + 4\mathbf{e_1}$ a $\mathbf{B_2} = 6\mathbf{e_1 e_2} + \mathbf{e_2}$. Pak jejich vnitřní součin získáme následovně.

$$\begin{aligned} \mathbf{B_1} \cdot \mathbf{B_2} = (2\mathbf{e_1 e_2} + 4\mathbf{e_1}) \cdot (6\mathbf{e_1 e_2} + \mathbf{e_2}) &= \langle (2\mathbf{e_1 e_2})(6\mathbf{e_1 e_2}) \rangle_{|2-2|} + \langle (2\mathbf{e_1 e_2})(\mathbf{e_2}) \rangle_{|2-1|} + \\ &+ \langle (4\mathbf{e_1})(6\mathbf{e_1 e_2}) \rangle_{|1-2|} + \langle (4\mathbf{e_1})(\mathbf{e_2}) \rangle_{|1-1|} = \\ &= 12\langle \mathbf{e_1 e_2 e_1 e_2} \rangle_0 + 2\langle \mathbf{e_1 e_2 e_2} \rangle_1 + \\ &+ 24\langle \mathbf{e_1 e_1 e_2} \rangle_1 + 4\langle \mathbf{e_1 e_2} \rangle_0 = \\ &= 12\langle -1 \rangle_0 + 2\langle \mathbf{e_1} \rangle_1 + 24\langle \mathbf{e_2} \rangle_1 + 4\langle \mathbf{e_1 e_2} \rangle_0 = \\ &= -12 + 2\mathbf{e_1} + 24\mathbf{e_2} \end{aligned}$$

Nyní obdobně zavedeme vnější součin multivektorů. Začneme opět bázovými blady a následně rozšířením na multivektory jako výše.

Definice 14 (Vnější součin geometrické algebry)**.** Nechť $\mathbf{E_1}, \mathbf{E_2}$ jsou bázové blady $\mathbb{G}_{p,q}$ a nechť $gr(\mathbf{E_1}) = k$, $gr(\mathbf{E_2}) = l$. Potom *vnější součin* dvou bázových bladů se definuje jako

$$\mathbf{E_1} \wedge \mathbf{E_2} := \langle \mathbf{E_1 E_2} \rangle_{k+l},$$

a pro multivektory $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{G}_{p,q}$ má následující vlastnosti

$$\begin{aligned} \mathbf{A} \wedge (\mathbf{B} + \mathbf{C}) &= \mathbf{A} \wedge \mathbf{B} + \mathbf{A} \wedge \mathbf{C}, \\ (a\mathbf{A}) \wedge (b\mathbf{B}) &= ab(\mathbf{A} \wedge \mathbf{B}), \end{aligned}$$

kde $a, b \in \mathbb{R}$.

Tedy vnější součin dvou multivektorů $\mathbf{A}, \mathbf{B} \in \mathbb{G}_{p,q}$, kde $\mathbf{E_k}$ jsou bázové blady, $a_k, b_k \in \mathbb{R}$, $\mathbf{A} = \sum_k a_k \mathbf{E}_k$ a $\mathbf{B} = \sum_k b_k \mathbf{E}_k$, lze spočítat

$$\mathbf{A} \wedge \mathbf{B} = \sum_{k,l} a_k b_l (\mathbf{E}_k \wedge \mathbf{E}_l).$$

Nyní znovu uveďme příklad použití. Záměrně použijeme stejné blady jako v předchozím příkladě.

Příklad 7. Nechť $\mathbf{B_1}, \mathbf{B_2} \in \mathbb{G}_{2,0}$, například $\mathbf{B_1} = 2\mathbf{e_1 e_2} + 4\mathbf{e_1}$ a $\mathbf{B_2} = 6\mathbf{e_1 e_2} + \mathbf{e_2}$. Pak jejich vnější součin získáme následovně

$$\begin{aligned} \mathbf{B_1} \wedge \mathbf{B_2} = (2\mathbf{e_1 e_2} + 4\mathbf{e_1}) \wedge (6\mathbf{e_1 e_2} + \mathbf{e_2}) &= \langle (2\mathbf{e_1 e_2})(6\mathbf{e_1 e_2}) \rangle_{2+2} + \langle (2\mathbf{e_1 e_2})(\mathbf{e_2}) \rangle_{2+1} + \\ &\quad \langle (4\mathbf{e_1})(6\mathbf{e_1 e_2}) \rangle_{1+2} + \langle (4\mathbf{e_1})(\mathbf{e_2}) \rangle_{1+1} = \\ &= 12\langle \mathbf{e_1 e_2 e_1 e_2} \rangle_4 + 2\langle \mathbf{e_1 e_2 e_2} \rangle_3 + 24\langle \mathbf{e_1 e_1 e_2} \rangle_3 + \\ &\quad + 4\langle \mathbf{e_1 e_2} \rangle_2 = 4\mathbf{e_1 e_2}. \end{aligned}$$

Nyní ukažme i geometrický součin dvou prvků z předchozích dvou příkladů.

Příklad 8. Nechť $\mathbf{B_1}, \mathbf{B_2} \in \mathbb{G}_{2,0}$, například $\mathbf{B_1} = 2\mathbf{e_1 e_2} + 4\mathbf{e_1}$ a $\mathbf{B_2} = 6\mathbf{e_1 e_2} + \mathbf{e_2}$. Pak jejich geometrický součin získáme následovně

$$\begin{aligned} \mathbf{B_1 B_2} = (2\mathbf{e_1 e_2} + 4\mathbf{e_1})(6\mathbf{e_1 e_2} + \mathbf{e_2}) &= 2\mathbf{e_1 e_2} 6\mathbf{e_1 e_2} + 2\mathbf{e_1 e_2 e_2} + 4\mathbf{e_1} 6\mathbf{e_1 e_2} + 4\mathbf{e_1 e_2} = \\ &= -12 + 2\mathbf{e_1} + 24\mathbf{e_2} + 4\mathbf{e_1 e_2}. \end{aligned}$$

Dále poznamenejme, že pro různé bázové vektory $\mathbf{e_i}, \mathbf{e_j} \in \mathbb{G}_{p,q}$ platí $\mathbf{e_i e_j} = \mathbf{e_i} \wedge \mathbf{e_j}$. Wedge bázových bladů používáme častěji i v bázi, jelikož je z něj zřejmá gradovaná struktura geometrické algebry. Nyní zaveďme další operátory, které budeme následně používat při konstrukci algoritmů.

2.2.3 Duál a reverze

Nyní uvedeme operaci, která se používá při aplikaci transformací a před ní zavedeme ještě definici inverze multivektoru.

Definice 15 (Inverze multivektoru)**.** Buďte multivektory $\mathbf{A}, \mathbf{B} \in \mathbb{G}_{p,q}$. Řekneme, že multivektor $\mathbf{B}$ je inverzí multivektoru $\mathbf{A}$, jestliže platí

$$\mathbb{AB} = 1.$$

Inverzi multivektoru $\mathbf{A}$ budeme značit $\mathbf{A^{-1}}$.

Definice 16 (Reverze)**.** [11]. Nechť $\mathbf{B} \in \mathbb{G}_{p,q}$ a n je stupeň pseudoskaláru tohoto prostoru. Pak reverzi definujeme

$$\tilde{\mathbf{B}} := \sum_{k=0}^{n} (-1)^{\frac{1}{2}k(k-1)} \langle \mathbf{B} \rangle_k.$$

Poznamenejme, že tato operace má následující vlastnosti

$$\widetilde{(\widetilde{\mathbf{B}})} = \mathbf{B} \text{ a } \widetilde{(\mathbf{B_1} \wedge \mathbf{B_2})} = \widetilde{\mathbf{B_1}} \wedge \widetilde{\mathbf{B_2}}.$$

Definice 17 (Duál)**.** [13] Nechť $\mathbf{E}$ je bázový blade $\mathbb{G}_{p,q}$ a $\mathbf{I}$ jednotkový pseudoskalár tohoto prostoru. Pak duál definujeme jako

$$\mathbf{E}^* := \mathbf{E}\mathbf{I}^{-1},$$

kde $\mathbf{I}^{-1}$ je inverze pseudoskaláru.

2.2.4 OPNS a IPNS

Nyní zmíníme dva typy reprezentací podprostorů $\mathbb{R}^n$ pomocí prvků geometrické algebry.

Definice 18 (Prostor nulových vnějších součinů (OPNS))**.** *Prostor nulových vnějších součinů* respektive OPNS (*outer product null space*) bladu $\mathbf{B} \in \mathbf{G}_{p,q}$ stupně k, značený $\mathbb{NO}(\mathbf{B})$, je definován jako

$$\mathbb{NO}(\mathbf{B}) := \{\mathbf{u} \in \mathbb{R}^n \ : \ \mathbf{u} \wedge \mathbf{B} = 0\}.$$

Definice 19 (Prostor nulových vnitřních součinů (IPNS))**.** *Prostor nulových vnitřních součinů* respektive IPNS (*inner product null space*) bladu $\mathbf{B} \in \mathbf{G}_{p,q}$ stupně k, značený $\mathbb{NI}(\mathbf{B})$, je definován jako

$$\mathbb{NO}(\mathbf{B}) := \{\mathbf{u} \in \mathbb{R}^n \ : \ \mathbf{u} \cdot \mathbf{B} = 0\}.$$

Mezi těmito reprezentacemi lze snadno přecházet pomocí operace duálu, kterou jsme zavedli výše. Navíc později budeme geometrické objekty reprezentovat právě reprezentacemi v OPNS a IPNS.

Kapitola 3

2D Konformní geometrická algebra CGA2

Dále se budeme zabývat takzvanou 2D konformní geometrickou algebrou $\mathbb{G}_{3,1}$, jejíž předností je hlavně jednoduchá práce s přímkami a kružnicemi. Této vlastnosti následně využijeme při konstrukci matematického modelu robotického ramene. V následující kapitole budu čerpat především z knihy [9], která se CGA2 zabývá mnohem podrobněji. Bází $\mathbb{R}^2$ je $(\mathbf{e_1}, \mathbf{e_2})$ a generátory $\mathbb{G}_{3,1}$ jsou $\{\mathbf{e_1}, \mathbf{e_2}, \mathbf{e}_+, \mathbf{e}_-\}$. Kvadratickou formou této geometrické algebry je

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

V bázi tohoto prostoru nám přibyly 2 bázové vektory $\mathbf{e}_+, \mathbf{e}_-$, s kladnou a zápornou signaturou, což znamená, že

$$\mathbf{e}_+{}^2 = 1, \quad \mathbf{e}_-{}^2 = -1, \quad \mathbf{e}_+ \cdot \mathbf{e}_- = 0.$$

Provedením změny báze získáme vektory

$$\mathbf{e_0} = \frac{1}{2}(\mathbf{e}_- - \mathbf{e}_+), \qquad \mathbf{e}_\infty = \mathbf{e}_- + \mathbf{e}_+,$$

přičemž tyto vektory pro nás budou mít geometrický význam. Skutečně, $\mathbf{e_0}$ reprezentuje bod v počátku $\mathbb{R}^2$ a $\mathbf{e}_\infty$ reprezentuje bod v nekonečnu. Vektory $\mathbf{e}_\infty$ a $\mathbf{e_0}$ jsou takzvané *null-vektory*, tedy

$$\mathbf{e_0}{}^2 = \mathbf{e}_\infty{}^2 = 0.$$

Bázové blady tohoto prostoru jsou uvedeny v tabulce 3.1.

Stupeň	Název	Blady	Počet
0	Skalár	1	1
1	Vektor	$\mathbf{e_1},\ \mathbf{e_2},\ \mathbf{e}_\infty,\ \mathbf{e_0}$	4
2	Bivektor	$\mathbf{e_1} \wedge \mathbf{e_2},\ \mathbf{e_1} \wedge \mathbf{e}_\infty,\ \mathbf{e_1} \wedge \mathbf{e_0},\ \mathbf{e_2} \wedge \mathbf{e}_\infty,\ \mathbf{e_2} \wedge \mathbf{e_0},\ \mathbf{e}_\infty \wedge \mathbf{e_0}$	6
3	Trivector	$\mathbf{e_1} \wedge \mathbf{e_2} \wedge \mathbf{e}_\infty,\ \mathbf{e_1} \wedge \mathbf{e_2} \wedge \mathbf{e_0},\ \mathbf{e_1} \wedge \mathbf{e}_\infty \wedge \mathbf{e_0},\ \mathbf{e_2} \wedge \mathbf{e}_\infty \wedge \mathbf{e_0}$	4
1	Pseudoskalár	$\mathbf{e_1} \wedge \mathbf{e_2} \wedge \mathbf{e}_\infty \wedge \mathbf{e_0}$	1

Tabulka 3.1: Bázové blady

Tato algebra má tedy dimenzi $2^4 = 16$.

3.1 Objekty CGA2

3.1.1 Bod

Body z $\mathbb{R}^2$ je v CGA2 třeba získat pomocí zobrazení $\mathcal{C}$ na nulový kužel, bližší informace se dají najít v [8, 12, 14]. Toto zobrazení vypadá následovně

$$\mathcal{C}(\mathbf{x}) = P := \mathbf{x} + \frac{1}{2}(\mathbf{x} \cdot \mathbf{x})\mathbf{e}_\infty + \mathbf{e_0},$$

kde $\mathbf{x} \in \mathbb{R}^2$. Vložením bodu $\mathbf{x} \in \mathbb{R}^2$ je tedy vektor $P \in \mathbb{G}_{3,1}$.

3.1.2 Přímka

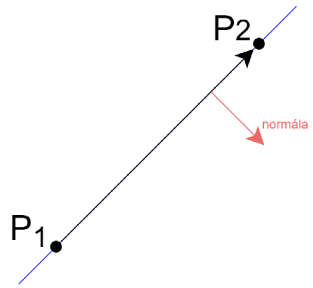
IPNS reprezentace přímky, tedy PNS reprezentace bodů $P = \mathbf{x} + \frac{1}{2}(\mathbf{x} \cdot \mathbf{x})\mathbf{e}_\infty + \mathbf{e_0}$ ležících v $\mathbb{R}^2$ na přímce L, je v CGA2 dána následovně

$$L = \mathbf{n} + d\mathbf{e}_\infty,$$

kde $\mathbf{n}$ je normálový vektor přímky L z $\mathbb{R}^2$ a $d \in \mathbb{R}$ je ortogonální vzdálenost bodu od počátku. V OPNS reprezentaci je konstrukce přímky intuitivnější. Lze si ji představit jako kružnici s nekonečným poloměrem, lze vidět z (3.2), procházejícími dvěma body P_1 a P_2 (a nebo jako spojnici dvou bodů s bodem v nekonečnu), tedy

$$L^* = P_1 \wedge P_2 \wedge \mathbf{e}_\infty.$$

Dále poznamenejme, že se skutečně jedná o kružnici s nekonečným poloměrem, což má za následek nepříjemnosti při výpočtu průniku, a to takové, že průnik dvou přímek není jeden, jak bychom přirozeně uvažovali, nýbrž jsou 2. Jeden v nekonečnu a druhý reálný, který bychom obyčejně očekávali.
Při konstrukci algoritmů a jejich ladění v praktické části této bakalářské práce jsme zaznamenali následující pozorování. Sestrojíme-li přímku v OPNS reprezentaci tímto způsobem $P_1 \wedge P_2 \wedge \mathbf{e}_\infty$, následně ji převedeme do IPNS pomocí operace duálu a vyčteme ze zápisu její normálu, tak tato normála má vždy jeden určitý směr i když jsou přípustné dva. Pro uchopitelnější představu si lze přestavit, že jdeme z bodu P_1 do bodu P_2, tedy v pořadí, ve kterém byly v konstrukci přímky. Pak normála této přímky(modrá) bude směřovat doprava od této přímky, jak je vidět na obrázku 3.1.



Obrázek 3.1: Orientace normály

Toto pozorování jsem zformuloval do následující věty.

Věta 20. *Nechť* $P_1 = \mathcal{C}(u\mathbf{e_1} + v\mathbf{e_2})$ *a* $P_2 = \mathcal{C}(x\mathbf{e_1} + y\mathbf{e_2})$ *jsou 2 rozdílné reálné body z* $\mathbb{R}^2$ *zobrazené do* $\mathbb{G}_{3,1}$ *a nechť* $L^* = P_1 \wedge P_2 \wedge \mathbf{e}_\infty$ *je OPNS reprezentace přímky. Pak* $L = L^*\widetilde{\mathbf{I}} = \mathbf{n} + d\mathbf{e}_\infty$, *má vektor normály* $\mathbf{n}$ *a koeficient* d *ve tvaru*

$$\begin{aligned}\mathbf{n} &= (y - v)\mathbf{e_1} + \mathbf{e_2}(u - x),\\ d &= (uy - vx).\end{aligned}$$

Důkaz. Nejdříve zobrazme body P_1 a P_2 do $\mathbb{G}_{3,1}$. Tedy

$$P_1 = \mathcal{C}(u\mathbf{e_1} + v\mathbf{e_2}) = u\mathbf{e_1} + v\mathbf{e_2} + \frac{u^2 + v^2}{2}\mathbf{e}_\infty + \mathbf{e_0},$$

$$P_2 = \mathcal{C}(x\mathbf{e_1} + y\mathbf{e_2}) = x\mathbf{e_1} + y\mathbf{e_2} + \frac{x^2 + y^2}{2}\mathbf{e}_\infty + \mathbf{e_0}.$$

Nyní sestrojme přímku L.

$$L^* = (u\mathbf{e_1} + v\mathbf{e_2} + \frac{u^2 + v^2}{2}\mathbf{e}_\infty + \mathbf{e_0}) \wedge (x\mathbf{e_1} + y\mathbf{e_2} + \frac{x^2 + y^2}{2}\mathbf{e}_\infty + \mathbf{e_0}) \wedge \mathbf{e}_\infty =$$

$$= (u\mathbf{e_1} + v\mathbf{e_2} + \frac{u^2 + v^2}{2}\mathbf{e}_\infty + \mathbf{e_0}) \wedge (x\mathbf{e_1} \wedge \mathbf{e}_\infty + y\mathbf{e_2} \wedge \mathbf{e}_\infty + \frac{x^2 + y^2}{2}\mathbf{e}_\infty \wedge \mathbf{e}_\infty + \mathbf{e_0} \wedge \mathbf{e}_\infty) =$$
$$= uy\mathbf{e_1}\wedge\mathbf{e_2}\wedge\mathbf{e}_\infty + u\mathbf{e_1}\wedge\mathbf{e_0}\wedge\mathbf{e}_\infty + vx\mathbf{e_2}\wedge\mathbf{e_1}\wedge\mathbf{e}_\infty + v\mathbf{e_2}\wedge\mathbf{e_0}\mathbf{e}_\infty + x\mathbf{e_0}\wedge\mathbf{e_1}\wedge\mathbf{e}_\infty + y\mathbf{e_0}\wedge\mathbf{e_2}\wedge\mathbf{e}_\infty.$$

A nyní provedeme operaci duálu, tedy převedeme přímku do IPNS reprezentace.

$$\begin{aligned}L = &(uy\mathbf{e_1} \wedge \mathbf{e_2} \wedge \mathbf{e}_\infty + u\mathbf{e_1} \wedge \mathbf{e_0} \wedge \mathbf{e}_\infty + vx\mathbf{e_2} \wedge \mathbf{e_1} \wedge \mathbf{e}_\infty + v\mathbf{e_2} \wedge \mathbf{e_0}\mathbf{e}_\infty +\\ &+ x\mathbf{e_0} \wedge \mathbf{e_1} \wedge \mathbf{e}_\infty + y\mathbf{e_0} \wedge \mathbf{e_2} \wedge \mathbf{e}_\infty)I^{-1} =\\ &= uy\mathbf{e}_\infty + u\mathbf{e_2} - vx\mathbf{e}_\infty - v\mathbf{e_1} - x\mathbf{e_2} + y\mathbf{e_1} =\\ &= (y - v)\mathbf{e_1} + (u - x)\mathbf{e_2} + (uy - vx)\mathbf{e}_\infty\end{aligned}$$

Odtud tedy plyne, že

$$\mathbf{n} = (y - v)\mathbf{e_1} + (u - x)\mathbf{e_2} \quad \text{a} \quad d = (uy - vx).$$

□

3.1.3 Kružnice

Pro reprezentaci kružnice C v IPNS potřebujeme bod $S = \mathcal{C}(\mathbf{x})$, střed kružnice, a její poloměr r. Reprezentace kružnice v IPNS je následující, [9].

$$C = S - \frac{1}{2}r^2\mathbf{e}_\infty = \mathbf{x} + \frac{1}{2}(\mathbf{x}^2 - r^2)\mathbf{e}_\infty + \mathbf{e_0}. \tag{3.1}$$

Při IPNS reprezentaci používáme především první tvar, nicméně z druhého tvaru lze názorně vidět, že bod je kružnicí s nulovým poloměrem. Kružnice v OPNS reprezentaci vypadá následovně

$$C^* = P_1 \wedge P_2 \wedge P_3. \tag{3.2}$$

Přičemž C^* je kružnice procházející právě body $P_1,\ P_2,\ P_3$. Je zde vidět na první pohled spojitost s OPNS reprezentací přímky, která je také kružnicí procházející třemi body. V algoritmech budeme také používat ještě jednu konstrukci kružnice v IPNS reprezentaci

z [10]. K sestrojení této kružnice budeme potřebovat střed kružnice S a bod P, kterým prochází. Tato reprezentace vypadá následovně

$$C = P \cdot (S \wedge e_\infty)$$

Při řešení algoritmů v pozdější části této práce je tento tvar použit poměrně často, případně je použit základní tvar IPNS (3.1). Střed s poloměrem nebo bodem, kterým kružnice prochází, jsou totiž většinou jediné informace, které máme.

3.1.4 Dvojbod

Jedním z velice zvláštních prvků CGA2 je takzvaný dvojbod. Je definován jako průnik dvou kružnic. Roli průniku v CGA2 hraje vnější dvou IPNS reprezentací objektů [9]. Tedy

$$P_p = C_1 \wedge C_2,$$

kde C_1, C_2 jsou IPNS reprezentace dvou kružnic. Poznamenejme, že výsledkem je vždy dvojbod. V případě že kružnice mají reálný průnik, tak výsledkem jsou právě dva body tohoto průniku.
Samotný dvojbod je spíše výsledkem průniků, takže ho většinou nezadáváme, ale je výsledkem výpočtů. V rámci těchto výpočtů je tedy třeba i přijít na to, z jakých dvou bodů se tento dvojbod skládá, protože to není na první pohled hned zřejmé. K výpočtu nám slouží následující vzorec

$$P_{1,2} = (P_p^* \pm \sqrt{P_p^* \cdot P_p^*})(e_\infty \cdot P_p^*)$$

3.2 Programování výpočtů v pythonu

V následujících kapitolách už začneme využívat programů v jazyce python, abychom ukázali výpočty v geometrické algebře. Budeme využívat balíku *clifford*, který obsahuje výpočetní nástroje pro geometrické algebry a geometrické algebry jako takové. Je možné si v něm vytvořit svou vlastní geometrickou algebru, nicméně my už využijeme předdefinované CGA2. Značení v tomto balíku je uvedeno v následující tabulce 3.2.

CGA2	Značení v clifford	CGA2	Značení v clifford	CGA2	Značení v clifford
$\mathbf{e_1}, \mathbf{e_2}$	e1, e2	$\mathbf{e}_+, \mathbf{e}_-$	ep, en nebo e4, e5	$\mathbf{e_0}$	eo
$\mathbf{e}_\infty$	einf	$\cdot$	|	$\wedge$	^
g. součin	*	C^*	(C).dual()	$\tilde{u}$	u~
Bod [x,y]	up(x*e1 + y*e2)	$\langle \mathbf{B} \rangle_1$	B(1)	$\tilde{u}$	~u
Bod do $\mathbb{R}^2$	down(Bod)				

Tabulka 3.2: Python legenda

Dále budeme využívat funkci *normal()*, která normalizuje blade vůči eukleidovské normě. K vykreslení geometrických objektů využijeme balíku *pyganja* a další potřebné matematické funkce z balíku *numpy*. *Pyganja* je schopna vykreslit i pole, do kterého jsou uložené objekty, což se nám bude hodit při iteracích algoritmů.

Dále každý program níže začíná stejnou hlavičkou. nebude-li uvedeno jinak. Tuto hlavičku již nebudeme v příkladech psát a je následující

```
from numpy import*
from clifford import *
from clifford.g2c import *
from pyganja import GanjaScene, draw
import pyganja;
```

Všechny kódy k programům jsou součástí přílohy.

3.3 Transformace v CGA2

Jednou z hlavních výhod geometrické algebry je jednoduchá reprezentace transformací a jejich aplikace. Oproti maticím rotace a podobným objektům se jedná o velké zjednodušení.

3.3.1 Reflexe

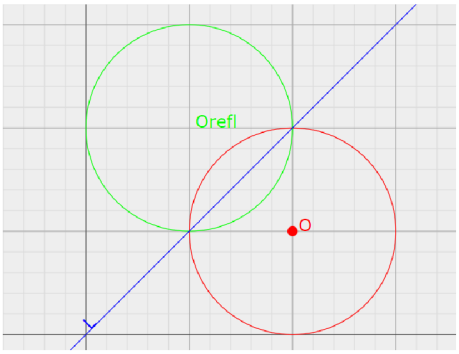
Reprezentace reflexe je vcelku přímočará. Mějme objekt $O \in \mathbb{G}_{3,1}$ a přímku L v IPNS reprezentaci procházející počátkem, tedy $L = \mathbf{n}$, přes kterou chceme provést reflexi objektu O. Příslušnou reflexi pak provedeme následovně.

$$O_{Refl} = -LOL.$$

Příklad 9. Buď O kružnice se středem $S = \mathcal{C}(2\mathbf{e_1} + \mathbf{e_2})$ a poloměrem 1, takže $O = S - \frac{1}{2}1^2\mathbf{e}_\infty$. Dále přímka L, procházející bodem $B = \mathcal{C}(\mathbf{e_1} + \mathbf{e_2})$ a počátkem. Pak reflexi kružnice O přes přímku L provedeme následovně.

```
S = up(2*e1+1*e2)
B = up(e1 + e2)
O = (S - 0.5 * 1 * 1 *einf).dual()
L = eo ^ B ^einf
Orefl = -L*O*L
GS = GanjaScene()
GS.add_object(S, color = (255, 0, 0))
GS.add_object(O, color = (255, 0, 0), label = 'O')
GS.add_object(Orefl, color = (0, 255, 0), label = 'Orefl')
GS.add_object(L, color = (0, 0, 255), label = 'L')
draw(GS,sig = layout.sig, scale = 0.6, grid = True)
```

Na obrázku 3.2 je následně vidět výsledek.



Obrázek 3.2: Reflexe

3.3.2 Rotace

Chceme-li provést rotaci objektu O okolo osy dané bivektorem $\mathbf{e_1} \wedge \mathbf{e_2}$,tedy okolo počátku, o úhel θ, pak rotor R konající tuto transformaci má tvar, [9],

$$R = \cos\left(\frac{\theta}{2}\right) - e_1 \wedge e_2 \sin\left(\frac{\theta}{2}\right). \tag{3.3}$$

Můžeme dále sestrojit obecný rotor následujícím způsobem. Chceme-li provést rotaci o úhel θ okolo bodu rotace B, pak rotor vypadá takto

$$R = \cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right)(B \wedge e_\infty)^*. \tag{3.4}$$

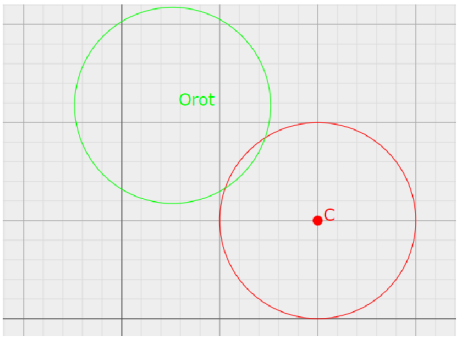
Poznamenejme, že při $B = e_0$, nám vznikne obyčejný rotor kolem počátku.

$$O_{rot} = RO\tilde{R}.$$

Příklad 10. Proveďme rotaci kružnice $C = S - \frac{1}{2}1^2\mathbf{e}_\infty$ kde $S = \mathcal{C}(2\mathbf{e1} + 1\mathbf{e2})$ o úhel $\theta = 50°$ okolo osy dané bivektorem $\mathbf{e_1} \wedge \mathbf{e_2}$.

```
S = up(2*e1+1*e2)
C = (S - 0.5 * 1 * 1 *einf).dual()
theta = (50 / 180) * pi;
R = cos(theta/2) - (e1 ^ e2) * sin(theta/2)
Orot = R * C * ~R
GS = GanjaScene()
GS.add_object(S, color = (255, 0, 0))
GS.add_object(C, color = (255, 0, 0), label = 'C')
GS.add_object(Orot, color = (0, 255, 0), label = 'Orot')
draw(GS,sig = layout.sig, scale = 0.65, grid = True)
```

Výsledek je uveden na obrázku 3.3.



Obrázek 3.3: Rotace

3.3.3 Translace

Translátor T pro translaci objektu O o vektor $\mathbf{t} = (t_1\mathbf{e_1} + t_2\mathbf{e_2} + t_3\mathbf{e_3})$ se sestrojí následovně [9],

$$T = 1 - \frac{1}{2}\mathbf{t}e_\infty$$

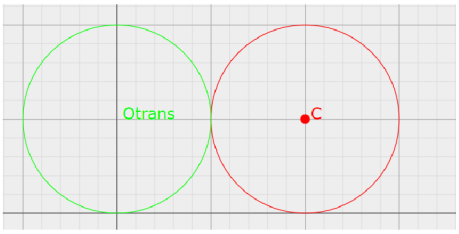
a aplikujeme jej takto

$$O_{tran} = TO\tilde{T}.$$

Příklad 11. Proveďme translaci kružnice $C = S - \frac{1}{2}1^2\mathbf{e}_\infty$, kde $S = \mathcal{C}(2\mathbf{e_1} + 1\mathbf{e_2})$ o vektor $\mathbf{t} = -2\mathbf{e_1}$.

```
S = up(2*e1+1*e2)
C = (S - 0.5 * 1 * 1 *einf).dual()
t = -2*e1
T = 1 - 0.5 * t * einf
Otrans = T * C * ~T
GS = GanjaScene()
GS.add_object(S, color = (255, 0, 0))
GS.add_object(C, color = (255, 0, 0), label = 'C')
GS.add_object(Otrans, color = (0, 255, 0), label = 'Otrans')
draw(GS,sig = layout.sig, scale = 0.6, grid = True)
```

Výsledek je uveden na obrázku 3.4.



Obrázek 3.4: Translace

3.3.4 Motor - obecný pohyb objektu

Pohyb hmotného tělesa, v angličtině označován *rigid body motion*, se provádí pomocí složení rotace R a translace T. Výsledným operátorem je *motor* M, který v geometrické algebře má tvar

$$M = TR\tilde{T} \tag{3.5}$$

a aplikujeme jej na objekt O, stejně jako translátor a nebo rotor, tedy

$$O_M = MO\tilde{M}.$$

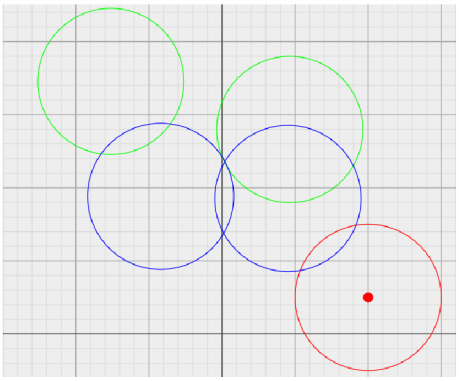
Příklad 12. Mějme kružnici C_1 se středem $S = \mathcal{C}(2\mathbf{e_1} + 0.5\mathbf{e_2})$ a s poloměrem $r_1 = 1$ (červená kružnice). Zvolme úhel $\theta = 50°$. Motor M sestrojíme podle (3.5). Pro sestrojení translátoru použijeme vektor $\mathbf{t} = -1\mathbf{e_1} + 0.5\mathbf{e_2}$. Kružnici C_1 následně dvakrát transformujeme pomocí motoru M (zelené kružnice). Pro kontrast kružnici C_1 také dvakrát otočíme pomocí rotoru, který jsme použili ke konstrukci motoru (modré kružnice). Příklad provedeme následujícím kódem.

```
'''Motor'''
S = up(2 * e1 + 0.5 * e2)
t = -e1 + 0.5 * e2
r_1 = 1
C_1 = (S - 0.5 * r_1 * r_1 *einf).dual()
theta = (pi / 180) * 50
R = cos(theta / 2) - (e1 ^ e2) * sin(theta / 2)
T = 1 - 0.5 * (t) * einf
M = T * R * ~T

C_1m = M * C_1 * ~M #Zelena
C_2m = M * C_1m * ~M
C_1r = R * C_1 *~R #Modra
C_2r = R * C_1r *~R

GS = GanjaScene()
GS.add_object(S, color = (255, 0, 0))
GS.add_object(C_1, color = (255, 0, 0))
GS.add_object(C_1m, color = (0, 255, 0))
GS.add_object(C_2m, color = (0, 255, 0))
GS.add_object(C_1r, color = (0, 0, 255))
GS.add_object(C_2r, color = (0, 0, 255))
draw(GS,sig = layout.sig, scale = 0.6, grid = True)
```

Výsledek je uveden na obrázku 3.5



Obrázek 3.5: Motor

3.4 Signatura trojúhelníku

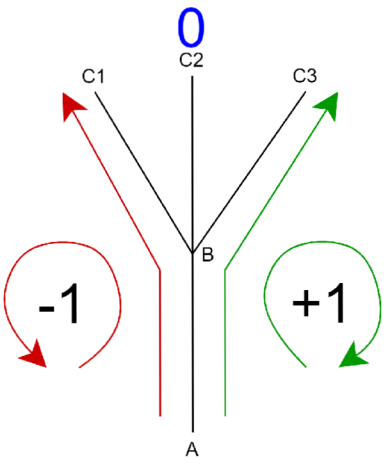
V této části budeme čerpat především z [10]. V algoritmech, které budeme následovně konstruovat budeme často potřebovat nástroj, který budeme nazývat *signatura trojúhelníku*. Tato vlastnost se nám bude hodit v algoritmech k zjištění informace o natočení robota nebo o vzhledu trasy zadané uživatelem do algoritmů. Mějme tři body $A, B, C \in \mathbb{G}_{3,1}$, po kterých se budeme chtít pohybovat, respektive je v nějakém pořadí projít. Pokud projdeme body podle abecedy, pak signaturou trojúhelníku rozumíme číslo

$$sign((A \wedge B \wedge C \wedge e_\infty)^*).$$

Toto číslo může nabývat tří hodnot, které nám říkají následující informaci

- -1, body jsme prošli proti směru hodinových ručiček
- 1, body jsme prošli po směru hodinových ručiček
- 0, body ležely na přímce

Pro představu si uveďme obrázek 3.6 se třemi možnými polohami bodu C, značenými C_1, C_2 a C_3, pro názornost toho jak poloha bodů ovlivní signaturu trojúhelníku.



Obrázek 3.6: Signatura

Ve skutečnosti se jedná o orientovaný objem. V 2D nám objem reálně vzniknout nemůže, nicméně si můžeme použít informaci o jeho orientaci.

Nyní už máme všechny potřebné informace k sestrojení algoritmů. Pokud by čtenáře problematika teorie geometrických algeber a aplikací zajímala více, lze nahlédnout také do [4, 2, 6, 7].

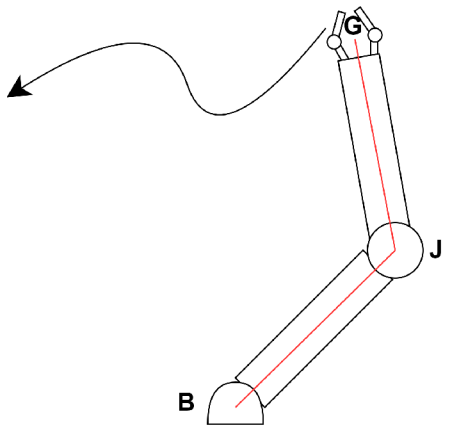
Kapitola 4

Inverzní kinematika a algoritmy

Jedním z cílů této bakalářské práce je řešit inverzní kinematiku, konkrétně se budeme zabývat inverzní kinematikou robotického ramene. Kinematika je odvětví mechaniky, které se obecně zabývá popisem pohybu. Klasický kinematický problém spočívá v otázce, jak se bude pohybovat těleso, respektive po jaké trajektorii, v případě daného zrychlení, polohy rychlosti apod. Inverzní kinematický problém spočívá v přesném opaku, jakými rychlostmi, zrychleními, v našem případě pohyby dosáhneme předem dané trajektorie nějakého hmotného bodu.

4.1 Model robota

V této bakalářské práci se budeme zabývat především modelovým příkladem robotického ramene v rovině, které má jeden pevný bod B (*base*) jako základnu, jeden bod J (*joint*) kloub a jeden bod G (*gripper*), reprezentující nástroj robota. Tyto body jsou spojeny takzvanými *linky*, které většinou nebudeme zobrazovat, zmiňujeme je především proto, že definují vzdálenosti bodů B, J a G. Tento model ukazuje obrázek číslo 4.1.



Obrázek 4.1: Model robota

Dále modelový robot nebude mít omezené rotace v kloubech. Inverzní kinematika pro tohoto robota se skládá z následující otázky, jak musíme pohybovat bodem J a G, tak aby trajektorie gripperu byla dle našich požadavků, respektive aby gripper vykonal konkrétní pohyb.

K řešení konkrétních typů trajektorií gripperu jsme sestrojili algoritmy a následně provedli jejich implementaci v již ve výše zmíněném programovacím jazyce *python* za pomocí balíků *clifford*, *pyganja* a *numpy*. Algoritmy jsme programovali v prostředí jupyter notebook, které je velice užitečné pro jednoduché výpočty a ladění algoritmů. Proto následující algoritmy postrádají strukturu funkcí z objektově orientovaného programování. Dále poznamenejme, že algoritmy jsou naprogramovány bez ověření smysluplnosti vstupu od uživatele, proto při zadání nerealizovatelných pohybů nefungují, respektive nevrací smysluplné výsledky.
V následujících algoritmech nebudeme explicitně označovat, zda se jedná o body z $\mathbb{R}^2$ nebo jejich vložení do $\mathbb{G}_{3,1}$, nicméně z kontextu to bude zřejmé.

4.2 HVN algoritmus

První algoritmus, kterým se budeme zabývat, je algoritmus na výpočet polohy bodu J. Je nutné zmínit, že tento algoritmus není původní, jde jen o implementaci v pythonu. Jedná se o algoritmus z článku Notes on Planar Inverse Kinematics Based on Geometric Algebra, [10] a tomuto algoritmu budu v této práci říkat *HVN algoritmus*, podle jeho autorů Hrdina-Vašík-Návrat. Chceme-li ramenem otočit tak, aby se gripper dostal do bodu G, potřebujeme k tomu znát konečnou polohu jointu a také rotory k provedení tohoto pohybu. Uveďme nyní teoreticky tento algoritmus. Budeme postupovat v popisu algoritmu sestupně dle implementace v pythonu, aby byly čtenáři jasné kroky, které se dějí v implementaci. Nicméně jej uvedeme pouze teoreticky, kód už kvůli většímu rozsahu uvádět nebudeme a bude k dispozici v příloze. Aby bylo zřejmé, jaké reprezentace objektů používáme při dosazování do vzorců, budou zde prvky OPNS označovány operátorem duálu a prvky IPNS bez něj.

Vstup:

- Počáteční poloha, tj. body B_0, J_0 a G_0.
- Konečný bod G polohy gripperu.

Výstup:

- Hledaný bod J.
- Úhel otočení prvního linku α.
- Úhel linků ve výsledné pozici φ.
- Rotory R_1 a R_2 k provedení pohybu.
- Vizualizace.

Postup algoritmu:

Nejprve potřebujeme sestrojit kružnici

$$S_B = J_0 \cdot (B \wedge \mathbf{e}_\infty),$$

která má střed bod B a prochází bodem J_0. Tato kružnice vystihuje všechny možné polohy jointů J. Protože první link je pevně připevněn k bodu B, musí body J na této kružnici

ležet. Následně potřebujeme zjistit délku druhého linku, a to provedeme tak, že sestrojíme kružnici $S_{G_0} = J_0 \cdot (G_0 \wedge \mathbf{e}_\infty)$ a zjistíme její poloměr r_{G_0} pomocí vzorce, [10].

$$r_{G_0} = \sqrt{S^*_{G_0} \cdot S^*_{G_0}}.$$

V implementaci je třeba také použít uvnitř odmocniny funkci *double()*, protože i když je výsledkem skalár, objektově se stále jedná o multivektor stupně 0 a ten funkce *sqrt()* neumí zpracovat. Toto bude potřeba při všech použitích odmocnin, ve kterých vystupují multivektory. Následně sestrojíme kružnici $S_G = G - \frac{1}{2} r^2_{G_0} \mathbf{e}_\infty$ se středem G a poloměrem r_{G_0}. Průnik S_G a S_B, nám dá 2 možné polohy jointu J, J_1 a J_2. Tento průnik provedeme pomocí vnějšího součinu IPNS reprezentací kružnic S_G a S_B. Tedy

$$J_1 \wedge J_2 = S_B \wedge S_G.$$

Pro rozložení dvojbodu budeme potřebovat následující vzorec, který následně budeme používat i v dalších algoritmech, [10].

$$J_{+,-} = (J_1 \wedge J_2 \pm \sqrt{(J_1 \wedge J_2) \cdot (J_1 \wedge J_2)})(e_\infty (J_1 \wedge J_2)) \tag{4.1}$$

Vypočtené body J_1 a J_2 budeme indexovat podle toho, jaké znaménko jsme použili ve vzorci, pokud to bude třeba. Nyní je třeba zjistit, který z dvojice bodů je hledaným bodem J. I když se jedná o modelový manipulátor, je zřejmé, že je nežádoucí, aby se při pohybu přetočil. Ať už z pohledu technického, že tento pohyb by nejspíše neměl být možný, a nebo kinematického, že kdyby se rameno při pohybu i prohnulo, tak by se pohybovalo po delší trajektorii. Zde použijeme právě signaturu trojúhelníku. Budeme potřebovat signaturu sig_0 trojúhelníku B, J_0 a G_0 v tomto pořadí. Použijeme k tomu funkci *sign*, která vrací hodnoty -1 nebo 1, podle znaménka argumentu. Tedy

$$sig_0 = sign((B \wedge J_0 \wedge G_0)^*).$$

Vhodný bod J_+ nebo J_- následně zvolíme tak, že ve vzorci na jeho výpočet musí být znaménko stejné jako v signatuře trojúhelníku. Tedy pokud $sig_0 = 1$, pak $J = J_+$ a pokud $sig_0 = -1$, pak $J = J_-$. Algoritmus tedy takhle zjistí, jaký je ohyb ramene v původní poloze a zvolí ji tak, aby finální ohyb ramene byl souhlasný s původním. Poznamenejme, že funkce *sign*, podobně jako funkce *sqrt()*, neumí zpracovat multivektor, a proto je zde znovu zapotřebí v kódu použít i funkci *double()*.
Následně algoritmus spočítá úhly α a φ. K výpočtu budeme potřebovat zkonstruovat normalizované přímky. Normalizace zde probíhá stejně jako v případě klasických vektorů, jen se zde ve výrazech nachází i blady. Analogicky budeme pracovat s koeficienty u bladů jako s koeficienty u vektorů a znormalizujeme je dle eukleidovské normy. V balíku *Clifford* na to slouží funkce *normal()*, kterou lze aplikovat na objekty typu multivektor. Budeme potřebovat sestrojit následující přímky.

$$\begin{aligned} p_{BJ_0} &= B \wedge J_0 \wedge \mathbf{e}_\infty, \\ p_{BJ} &= B \wedge J \wedge \mathbf{e}_\infty, \\ p_{JB} &= J \wedge B \wedge \mathbf{e}_\infty, \\ p_{JG} &= J \wedge G \wedge \mathbf{e}_\infty. \end{aligned}$$

Úhly, které hledáme, následně získáme po zmíněné normalizaci přímek díky vlastnostem vnitřního součinu. Tedy například úhel pootočení prvního linku získáme jako

$$\alpha = \arccos\left(p_{BJ_0} \cdot p_{BJ}\right)$$

Znovu je zde potřeba i funkce *double()* a výsledek v programu navíc převádíme ještě na stupně, jelikož výsledkem této funkce je úhel v radiánech. Obdobně získáme úhel prohnutí

$$\varphi = \arccos\left(p_{JB} \cdot p_{JG}\right).$$

Nyní už zbývá získat rotory pro provedení pohybu, které budou realizovat výsledný pohyb. K tomu budeme potřebovat i pomocné body. Nejdříve získáme bod

$$J_C = (J_0 \wedge J)\mathbf{e}_\infty(J_0 \wedge J).$$

A z něj následně získáme rotor

$$R_1 = (B \wedge J_C \wedge \mathbf{e}_\infty)(B \wedge J_0 \wedge \mathbf{e}_\infty).$$

Máme tedy rotor R_1 pro rotaci bodu J. Jelikož se celé rameno pootočilo o úhel α po této rotaci, musíme otočit i bodem G, tedy $G_1 = R_1 G_0 \tilde{R}_1$. Rotor pro rotaci bodu G pak získáme analogicky, tedy

$$G_C = (G_1 \wedge G)e_\infty(G_1 \wedge G)$$

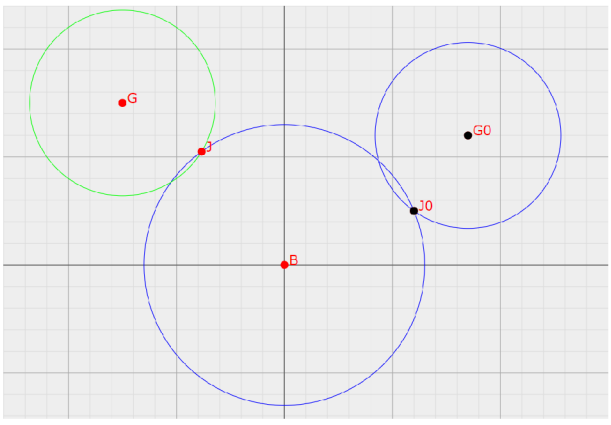
a následně

$$R_2 = (J \wedge G_C \wedge \mathbf{e}_\infty)(J \wedge G_1 \wedge \mathbf{e}_\infty).$$

Poznamenejme, že tyto rotory ponesou informaci o pohybu po kružnici, bod J se bude po kružnici pohybovat vždy, ale kdybychom potřebovali pohyb bodu G například po přímce, budeme potřebovat translátor. Tuto možnost uvedeme až v následujícím příkladu.
Dalším výstupem algoritmu je vizualizace v *paganja* pro kontrolu. Pro vstup

```
B = up(0)
J_0 = up(1.2 * e1 + 0.5 * e2)
G_0 = up(1.7 * e1 + 1.2 * e2)
G = up(-1.5 * e1 + 1.5 * e2)
```

je vizualizace na obrázku 4.2.



Obrázek 4.2: Algoritmus HVN

Je třeba dodat, že k názornější vizualizaci by bylo vhodnější, aby body robota byly spojeny úsečkami, nicméně tuto možnost balík *pyganja* pro 2D vizualizaci neumožňuje.

Tím algoritmus končí

Následující algoritmus budeme cyklicky používat pro série bodů pohybu, které určíme dalšími algoritmy. Dostaneme tedy body pohybu G a mezi posunem z jednoho do následujícího vždy provedeme tento algoritmus.
Stojí za zmínku, že následující algoritmus dopočtu polohy kloubů jako jediný závisí na konstrukci robota, respektive modelu robota. Zbylé algoritmy už jen počítají trasu gripperu, která na robotovi závisí jen z pohledu, zda je trasa gripperu dosažitelná, a to, že by ji robot nemohl provést, bychom se dozvěděli až z nesmyslného výsledku algoritmu na dopočet kloubů. Nese to s sebou nevýhodu, že můžeme provést velkou část výpočtu a následně až na konci zjistit, že počítáme pohyb, který nelze uskutečnit. Nicméně tento postup s sebou nese i značnou výhodu. Díky tomu, že algoritmus na dopočet kloubů je jediná část, která závisí na modelu robota, stačí k algoritmům na výpočet pohybu gripperu použít jiný algoritmus na dopočet kloubů a bude fungovat pro jiného robota. Následující algoritmy lze tedy použít i pro jiné roboty, než je náš modelový.

4.3 Pohyb po přímce

Druhým algoritmem, kterým se budeme zabývat, je algoritmus pohybu gripperu po přímce. Uvádíme ho dříve, protože je jednodušší než pohyb po kružnici, i když by se na první pohled mohlo zdát, že to bude naopak.

Vstup:

- Počáteční poloha, tj. body B_0, J_0 a G_0.
- Konečný bod G pohybu gripperu.
- Počet kroků pohybu n.

Výstup:

- Hledaný bod J.
- List bodů pohybu gripperu Gn.
- List bodů pohybu jointu Jn.
- Vizualizace.

Postup algoritmu:

Postup je ve své podstatě velice jednoduchý. Stačí získat z počátečního a koncového bodu pohybu translátor a vektor určující tento translátor následně rozdělit podle počtu kroků n. Potřebujeme tedy získat vektor $\mathbf{t}$, ten obdržíme ze souřadnic bodů G_0 a G v $\mathbb{R}^2$. K tomu, abychom je získali, lze programátorsky přistoupit dvěma způsoby. První variantou je získání složek u $\mathbf{e_1}$ a $\mathbf{e_2}$ pomocí hranatých závorek. Příkaz $G[e1]$ vrátí koeficient u složky $\mathbf{e_1}$. Tento příkaz lze použít i na všechny blady. My nicméně využijeme vlastností vnitřního součinu a tyto složky získáme jako vnitřní součiny bodů s bázovými vektory $\mathbf{e_1}$ a $\mathbf{e_2}$. Vektor $\mathbf{t}$ bude vypadat tedy následovně

$$\mathbf{t} = (G \cdot \mathbf{e_1} - G_0 \cdot \mathbf{e_1})\mathbf{e_1} + (G \cdot \mathbf{e_2} - G_0 \cdot \mathbf{e_2})\mathbf{e_2}.$$

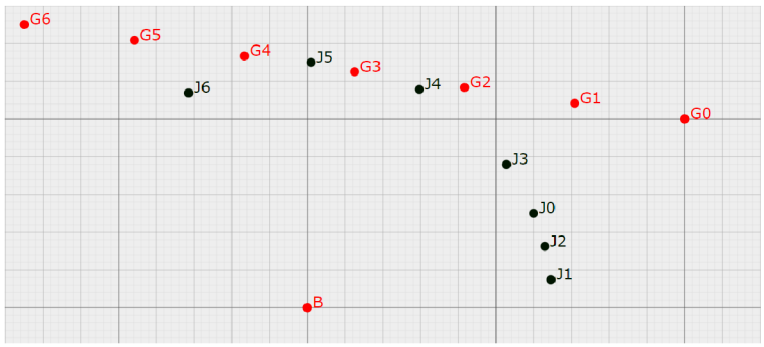
Tento vektor v programu nepočítáme zvlášť, nýbrž rovnou uvnitř translátoru, který sestrojíme, jak jsme si již řekli výše, následovně

$$T = 1 - \frac{1}{2}\frac{\mathbf{t}}{n}\mathbf{e}_\infty.$$

Následně vytvoříme list bodů pohybu Gn, kde n není číselný index, pro indexaci budeme používat čísla a index i. Tento list vytvoříme tak, že n-krát provedeme translaci bodu G_0 a všechny body po ní vždy uložíme do listu. Následně v cyklu provedeme opakovaně výpočet poloh kloubů J pomocí HVN algoritmu, přičemž v každé iteraci vypočtený bod J uložíme do listu Jn a použijeme jej k výpočtu bodu dalšího. Tento algoritmus cyklicky aplikujeme tak, že použijeme výchozí polohu robota a následující bod pohybu z listu Gn, a vypočteme k ní polohu bodu J. Tuto polohu nyní použijeme jako výchozí a vezmeme další bod z listu Gn a tento postup opakujeme, dokud nespočítáme polohy kloubů ke všem bodům z listu Gn. Je třeba dodat, že při realizaci translací vznikají v programu numerické chyby v podobě vlastností objektů na pozadí, které mohou ovlivnit výpočet nebo zabránit vykreslení. Tuto potíž obejdeme následujícím způsobem. Víme, že se jedná o bod, a tudíž pomocí funkce *down()*, která provede projekci bodu z $\mathbb{G}_{3,1}$ do $\mathbb{R}_2$, jej převedeme na bod $\mathbb{R}^2$. Následně tento bod znovu zobrazíme pomocí funkce *up()* a tím jej znovu inicializujeme. Tento styl vyčištění přebytečných vlastností bodu, které jsou na pozadí programu, se v programech objevuje velice četně, jelikož se jedná o častý problém. U bodů po transformaci je to skoro pravidlem. Následně výsledek vizualizujeme.
Pro vstup

```
n = 6
B = up(0)
J_0 = up(6 * e1 + 2.5 * e2)
G_0 = up(10 * e1 + 5 * e2)
G = up(-7.5 * e1 + 7.5 * e2)
```

je vizualizace na obrázku 4.3.



Obrázek 4.3: Algoritmus pohybu po přímce

Tím algoritmus končí

Tento algoritmus ještě následně dvakrát využijeme při složitějších pohybech. Nyní ukážeme další algoritmus, a to algoritmus pohybu po kružnici.

4.4 Pohyb po kružnici

Původní myšlenkou bylo provádět tento pohyb pomocí rotoru (3.3), jenomže to by mělo za následek následující dvě věci. Střed rotace bodu G_0 by musel ležet v počátku a za druhé by cílový bod musel ležet na kružnici $S = G_0 \cdot (B \wedge \mathbf{e}_\infty)$. To sice z matematického hlediska není žádný problém, ale fakt, že budeme mít tyto požadavky pro programátora nebo operátora robota, je absurdní, protože takových situací určitě není mnoho. Proto jsme zvolili složitější verzi, která se bude hodit i v dalším algoritmu a je mnohem praktičtější. Gripper se bude totiž pohybovat po kružnici se zadaným poloměrem a směrem pohybu. Tímto zadáním jsme se inspirovali v programování CNC strojů u funkcí G02 a G03. Výsledný pohyb tedy bude prováděn pomocí obecného rotoru (3.4).

Vstup:

- Počáteční poloha tj. body B_0, J_0 a G_0.
- Konečný bod G pohybu gripperu.
- Počet kroků pohybu n.
- Směr pohybu.
- Poloměr pohybu r.

Výstup:

- List bodů pohybu gripperu Gn.
- List bodů pohybu jointu Jn.
- Vizualizace.

Postup algoritmu:

K provedení pohybu po kružnici nejdříve potřebujeme znát střed tohoto pohybu. Ten algoritmus zjistí tak, že si sestrojí dvě kružnice S_{r_0} a S_r o poloměru r v počátečním a koncovém bodě. Tedy

$$S_{r_0} = G_0 - \frac{1}{2} r^2 \mathbf{e}_\infty,$$
$$S_r = G - \frac{1}{2} r^2 \mathbf{e}_\infty.$$

Následně z průniku kružnic S_{r_0} a S_r získáme dva body, kde může střed rotace ležet. Tento průnik, respektive dvojbod, získáme takto

$$P_p = S_{r_0} \wedge S_r.$$

Následně oba body dvojbodu, P_+ a P_-, získáme pomocí vzorce (4.1) na rozklad dvojbodu. Správný střed zvolíme na základě signatury trojúhelníku. Budeme-li chtít pohyb po kružnici, která je zakřivená směrem od základny B, zjednodušeně řečeno, pak signatura trojúhelníku G_0BG a G_0PG bude souhlasná. Tedy algoritmus vybere ten bod P, který

tuto podmínku splňuje. Kdybychom chtěli opak, že kružnice bude zakřivená směrem k základně, pak signatura trojúhelníku bude rozdílná, a algoritmus vybere ten bod, který tuto podmínku splňuje. K sestrojení motoru ještě potřebujeme úhel rotace. Úhel zjistíme z úhlu dvou přímek

$$p_{PG_0} = P \wedge G_0 \wedge \mathbf{e}_\infty,$$

$$p_{PG} = P \wedge G \wedge \mathbf{e}_\infty,$$

které normalizujeme a následně z nich pomocí vlastností vnitřního součinu získáme úhel rotace φ. Tedy

$$\varphi = \arccos\left(p_{PG_0} \cdot p_{PG}\right).$$

Nyní už jen stačí vhodně sestrojit obecný rotor pro provedení pohybu. V obecném rotoru zároveň dělíme úhel otočení počtem kroků, abychom pohyb rozdělili na požadovaný počet bodů listu Gn. V případě, že bude trajektorie zakřivená směrem od středu, zvolíme obecný rotor

$$R = \cos\left(\frac{-\varphi}{2n}\right) - \sin\left(\frac{-\varphi}{2n}\right)(P \wedge \mathbf{e}_\infty)^*,$$

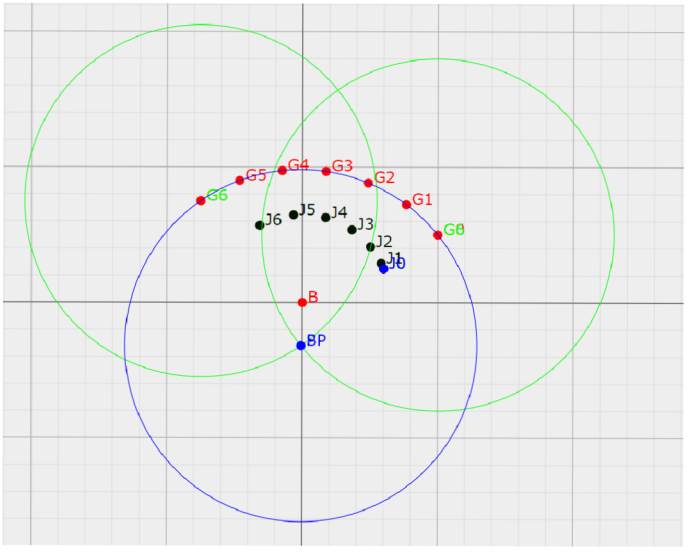
v opačném případě zvolíme obecný rotor

$$R = \cos\left(\frac{\varphi}{2n}\right) - \sin\left(\frac{\varphi}{2n}\right)(P \wedge \mathbf{e}_\infty)^*.$$

Následně získáme list bodů pohybu gripperu Gn cyklickým aplikováním obecného rotoru na počáteční bod G_0, dokud se nedostaneme do bodu G, tedy neprovedeme transformaci n-krát. Poté získáme list Jn poloh bodů kloubů J pomocí HVN algoritmu stejně jako v případě pohybu po přímce. Následně proběhne vizualizace. Pro vstup

```
B = up(0)
J_0 = up(0.6 * e1 + 0.25 * e2)
G_0 = up(1 * e1 + 0.5 * e2)
G = up(-0.75 * e1 + 0.75 * e2)
r =1.3
n = 6
G02 = 0 #Kruznice prohnuta smerem ke stredu
G03 = 1 #Kruznice prohnuta smerem od stredu
```

získáme obrázek 4.4.

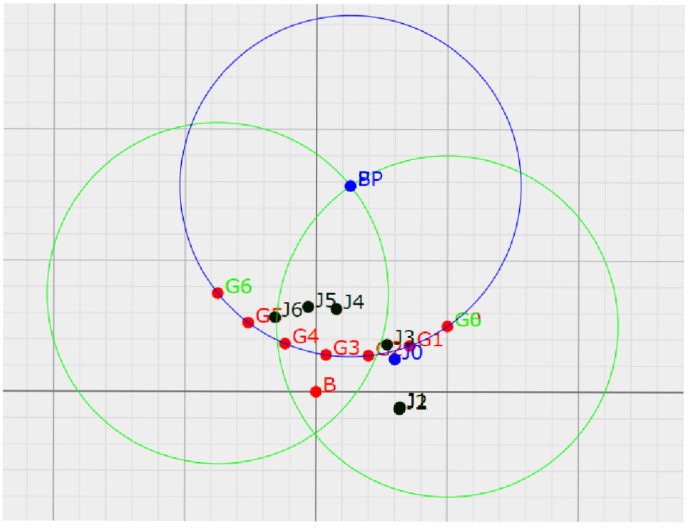


Obrázek 4.4: Algoritmus pohyb po kružnici G03

A pro vstup

```
B = up(0)
J_0 = up(0.6 * e1 + 0.25 * e2)
G_0 = up(1 * e1 + 0.5 * e2)
G = up(-0.75 * e1 + 0.75 * e2)
r =1.3
n = 6
G02 = 1 #Kruznice prohnuta smerem ke stredu
G03 = 0 #Kruznice prohnuta smerem od stredu
```

získáme obrázek 4.5.



Obrázek 4.5: Algoritmus pohyb po kružnici G02

Tím algoritmus končí

Obzvláště u tohoto algoritmu je třeba dát pozor na zadané hodnoty, protože jednoduše vzniknou body pohybu, které jsou mimo dosah robota.

4.5 Pohyb po lomené čáře

V technické praxi tento druh pohybu nalezneme v případě, kdy pohyb po křivce aproximujeme pomocí pohybu po lomené čáře. Vstupem je tedy posloupnost bodů, která může být buď dána uživatelem, nebo určena pomocí nějakého funkčního předpisu zadané křivky. Také tento pohyb může být užitečný, pokud se chce robot po cestě vyhnout určitým překážkám, ale to zmíníme na konci algoritmu. Tedy

Vstup:

- Počáteční poloha, tj. body B_0, J_0 a G_0.
- Body G, přes které povede pohyb gripperu (3 za sebou nesmí být na přímce).
- Počet kroků pohybu n mezi každými dvěma následujícími body.

Výstup:

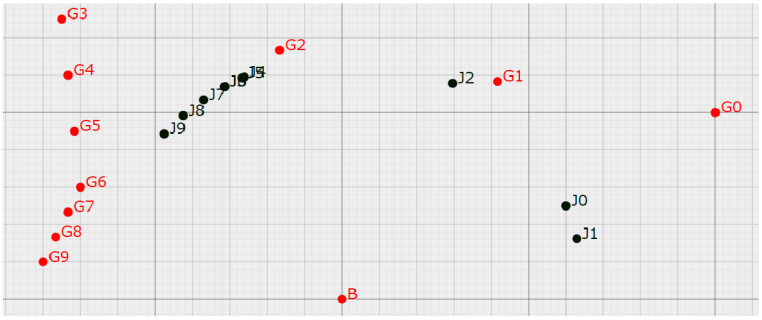
- List bodů pohybu gripperu Gn.
- List bodů pohybu jointu Jn.
- Vizualizace.

Postup algoritmu:

Jedná se pouze o cyklické použití algoritmu pohybu po přímce. Nejprve algoritmus v první iteraci provede výpočet bodů pohybu po přímce mezi počátečním bodem G_0 a prvním bodem z listu bodů G a uloží je do listu Gn. Následně ke všem těmto bodům pomocí HVN algoritmu zjistí polohy jointů J a ty uloží do listu Jn. V druhé iteraci se algoritmus posune na dvojici první bod z listu G a druhý bod z listu G. A počítá znovu body pohybu G_i a polohy jointů J. Dále se už jen analogicky posouvá v listu bodů G až nakonec. Nakonec program provede vizuální výstup. Pro vstup

```
B = up(0)
J_0 = up(6 * e1 + 2.5 * e2)
G_0 = up(10 * e1 + 5 * e2)
n = 3
G_list = [up(-7.5 * e1 + 7.5 * e2), up(-7 * e1 + 3 * e2),...
up(-8 * e1 + 1 * e2)]
```

získáme obrázek 4.6.



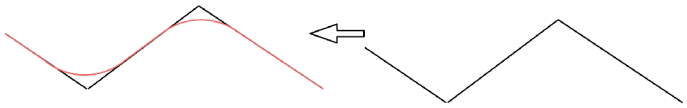
Obrázek 4.6: Algoritmus pohybu po lomené čáře

Tím algoritmus končí

Jak jsme již zmínili výše, tento algoritmus je vhodný k aproximaci trasy po křivce. Nicméně kdybychom jej chtěli použít v praxi k tomu, aby se robot vyhnul různým překážkám, tak vzdálenosti mezi body G by byly veliké jako na obrázku 4.6. Kolegové z praxe, kteří se i se mnou zabývají automatizací, mě upozornili na fakt, že pohyb robota po takové křivce nebude plynulý. Respektive v rozích lomené čáry bude muset robot zastavovat na nulovou rychlost, což má nepříznivý vliv na mechanizmus robota a může mít i špatný vliv na činnost, kterou vykonává. Převedeno do matematické řeči, poukazovali na fakt, že se nejedná o křivku třídy C_1. Zároveň jsem se od kolegů dozvěděl, že se tento problém řeší tak, že gripper se v rozích pohybuje po kružnicích, což je i výborné řešení, protože s kružnicemi umíme v CGA2 zacházet velice dobře. Proto vznik algoritmus následující.

4.6 Pohyb po lomené čáře prokládaný kružnicemi

Tento algoritmus je prakticky kombinací všech předchozích. Velkou výhodou je, že jsme vytvořili pohyb po kružnici tak, že je připravený na stejné vstupy jako budou zde. Respektive orientace pohybu po kružnici určí algoritmus, uživatel zadá jen poloměr. Změna trasy pak může například vypadat jako na obrázku 4.7.



Obrázek 4.7: Proložení lomené čáry

Vstup:

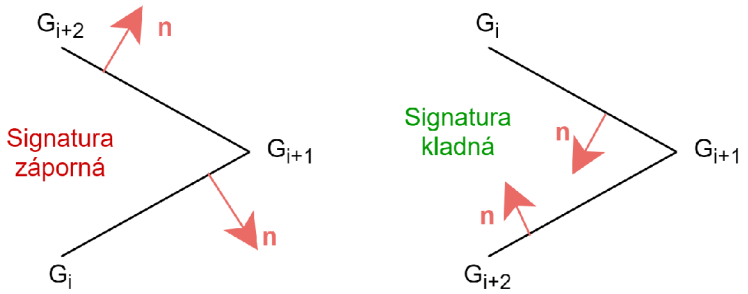
- Počáteční poloha tj. body B_0, J_0 a G_0.
- Body G, přes které povede pohyb gripperu (3 za sebou nesmí být na přímce).
- Počet kroků pohybu n mezi každými dvěma následujícími body.
- Poloměr kružnic r.

Výstup:

- List bodů pohybu gripperu Gn.
- List bodů pohybu jointu Jn.
- Vizualizace.

Postup algoritmu:

Na začátku výpočtu potřebuje algoritmus zjistit, kde mají ležet kružnice, po kterých se bude robot pohybovat, a potřebuje tedy informaci o tom, jak trasa bez kružnic po lomené čáře vypadá. Označme G_i, kde $i = 0, 1, ..., k$, body pohybu, kde G_0 je počáteční bod a G_1 až G_k jsou zadané body pohybu. Algoritmus sestrojí k-tici OPNS přímek procházejících body G_i a G_{i+1} pro $i = 0, 1, ..., k-1$. Nyní algoritmus pro každou trojici bodů G_i, G_{i+1} a G_{i+2} a dvojici přímek $L1 = G_i \wedge G_{i+1} \wedge \mathbf{e}_\infty$, $L_2 = G_{i+1} \wedge G_{i+2} \wedge \mathbf{e}_\infty$ provede následující. Spočítá signaturu trojúhelníku $G_iG_{i+1}G_{i+2}$. Tato signatura nám říká, jakým způsobem jsme body prošli, a díky Větě 20 také říká, kterým směrem jsou orientované normály, jako na obrázku číslo 4.8.



Obrázek 4.8: Algoritmus pohybu po lomené čáře

Poznamenejme, že je tedy zřejmě vidět, že pokud budeme mít zápornou signaturu, pak normály duálů přímek L_1 a L_2 budou směřovat vždy vně z trojúhelníku. Naopak, pokud bude signatura kladná, tak normály budou směřovat vždy směrem do trojúhelníku.
Nyní algoritmus nalezne střed kružnice tak, že sestrojí translátor a posune přímky L_1 a L_2 do vnitřní strany rohu pohybu. Tedy algoritmus zjistí normály přímek L_1 a L_2 pomocí operace duálu, viz Věta 20. Znormalizuje tyto normály dle eukleidovské normy a následně je vynásobí poloměrem kružnice r. Pokud byla signatura záporná, tak normály směřují vně z trojúhelníku, a tedy i opačným směrem než požadujeme, vynásobíme tedy navíc normálové vektory -1 při záporné signatuře. Tím jsme tedy zjistili vektory $\mathbf{t_1}$ z normály přímky L_1 a $\mathbf{t_2}$ z normály přímky L_2, potřebné pro translaci. Translátory T_1 pro L_1 a T_2 pro L_2 sestrojíme následovně

$$T_1 = 1 - \frac{1}{2}\mathbf{t_1}\mathbf{e}_\infty,$$

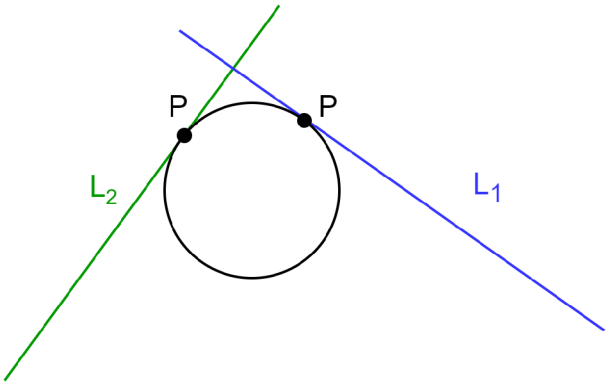
$$T_2 = 1 - \frac{1}{2}\mathbf{t_2}\mathbf{e}_\infty.$$

Nyní provedeme translaci přímek L_1 a L_2 příslušnými translátory. Tedy $L_{1trans} = T_1 L_1 \tilde{T_1}$ a $L_{2trans} = T_2 L_2 \tilde{T_2}$. Průsečíkem těchto dvou přímek je dvojbod $PP = L_{1trans} \wedge L_{2trans}$, který

následně rozložíme pomocí vzorce z [9]. Tedy dvojbod PP znormalizujeme vůči $\mathbf{e_1} \wedge \mathbf{e_2}$, následně koeficienty p_1 a p_2, kde p_1 je koeficient dvojbodu PP u bladu $\mathbf{e_2} \wedge \mathbf{e_3}$ vynásobený -1 a p_2 je koeficient dvojbodu PP u bladu $\mathbf{e_1} \wedge \mathbf{e_3}$. Průsečík P potom vypadá následovně

$$P = \mathcal{C}(p_1\mathbf{e_1} + p_2\mathbf{e_2}).$$

Body postupně ukládáme do listu. Takto si algoritmus pro každou trojici bodů, respektive pro každý roh, zjistí střed kružnice.
Nyní si v každém z těchto bodů P sestrojí kružnici o poloměru r, tyto kružnice si také ukládá do listu. Algoritmus nyní zjistí upravený tvar hlavních bodů pohybu G. Tyto body získá tak, že vytvoří list bodů, na jehož začátku je G_0. Následně popořadě pro každou trojici bodů najde průsečík P jejich kružnice, nejdříve s L_1 a následně s L_2 jako na obrázku 4.9, a v tomto pořadí je uloží.

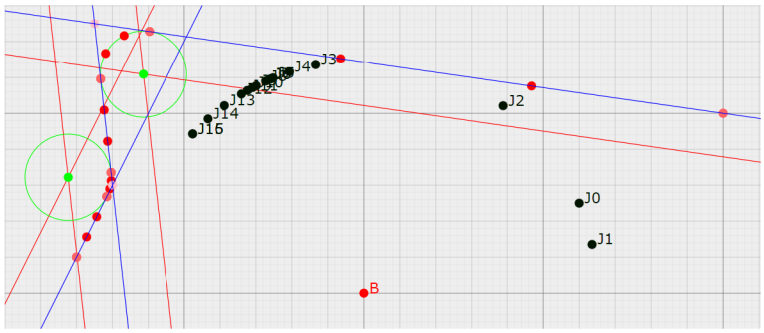


Obrázek 4.9: Průsečíky přímek a kružnic

Takto pokračuje pro všechny trojice a na závěr připojí bod G_k. Tím jsme dostali sérii bodů G, po kterých se budeme střídavě pohybovat po přímce a po kružnici, což již umíme. Algoritmus pro pohyb po kružnici již nečerpá vstup o směru pohybu, protože střed pro konstrukci obecného rotoru, čili středy kružnic ve zlomech pohybu, jsme již vypočítali a čerpáme z nich. Polohy kloubů získáme znovu cyklickým použitím algoritmu HVN.
Pro vstup

```
B = up(0) #Base poin
J_0 = up(6 * e1 + 2.5 * e2) #Joint point pocatecni
G_0 = up(10 * e1 + 5 * e2) #Efector point pocatecni
r = 1.2
n = 3 #pocet kroku
G_list = [G_0, up(-7.5 * e1 + 7.5 * e2),...
up(-7 * e1 + 3 * e2), up(-8 * e1 + 1 * e2)]
```

Jako výstup dostaneme následující obrázek 4.10.



Obrázek 4.10: Algoritmus pohybu po lomené čáře proložený kružnicemi

Tím algoritmus končí

Tímto jsme trasu po lomené čáře proložili trasami po kružnicích tak, že výsledná křivka je diferencovatelná. Poznamenejme, že jsme využili značného výpočetního zjednodušení díky znalosti signatury trojúhelníku a orientace normály z konstrukce přímek. Algoritmus tak nemusí provádět složitější výpočty, aby zjistil potřebné informace o tvaru trasy. Tento algoritmus je závěrečným algoritmem této práce.
V příloze lze nalézt také implementaci algoritmu z článku Radka Tichého [3] na dopočet kloubů reálného robota, jedná se o algoritmus podobného typu jako HVN. Tento algoritmus nebyl v práci popisován a byl naprogramován v rámci tréninku během rešerše, nicméně jej přikládáme jako ukázku.

Kapitola 5

Závěr

Cílem této práce bylo získání základních znalostí geometrických algeber a jejich následná aplikace na inverzní kinematický problém robotického ramene.
V první části bakalářské práce jsme se zabývali bilineárními a kvadratickými formami, které jsme následně použili ke konstrukci geometrické algebry. Teoretické pojmy byly doplněny názornými příklady pro vyjasnění postupů výpočtů a provedení konstrukcí vnitřního a vnějšího součinu. Vnitřní a vnější součin jsme následně rozšířili z vektorů na celou geometrickou algebru a zobecnili jejich výpočty pomocí projekcí. Byly také zavedeny operace reverze a duál a také prostorové reprezentace OPNS a IPNS.
V druhé části bakalářské práce jsme zavedli konformní geometrickou algebru, ukázali jsme konstrukci geometrických objektů v této algebře a tuto konstrukci jsme doplnili o vlastní Větu 20 o orientaci normály včetně důkazu, která byla stěžejní v následné aplikaci. Dále jsme čtenáře seznámili se základní prací s geometrickou algebrou v programovacím jazyce python. Díky tomu jsme mohli zavedení transformací doplnit i názornými ukázkami a příklady reflexe, translace, rotace a obecného pohybu. Na závěr této části jsme čtenáře seznámili s pojmem signatury trojúhelníku, o kterém jsme zjistili, že má v algoritmech velký potenciál.
V třetí části bakalářské práce jsme zavedli inverzní kinematický problém a vytvořili si modelový příklad robota, na kterém jsme následně testovali sestrojené algoritmy v jazyce python k řešení inverzního kinematického problému. Prvním algoritmem byl algoritmus HVN, který byl implementován v rámci rešerše. Následoval návrh již úplně vlastních algoritmů. Nejdříve jsme zkonstruovali algoritmus pro pohyb gripperu po přímce a následně pro pohyb po kružnici. Tyto algoritmy pak posloužily ke konstrukci komplexnějších algoritmů. Nejdříve se jednalo pouze o algoritmus pohybu po lomené čáře, ale na základě připomínek, které jsme obdrželi z praxe, jsme tuto práci nad rámec doplnili o algoritmus pohybu po lomené čáře prokládané kružnicemi. V rámci tohoto algoritmu jsme dosáhli pozorování, které vyústilo v již výše zmíněnou Větu 20, a toto pozorování bylo využito k podstatnému zjednodušení výpočtu v kombinaci se signaturou trojúhelníku.
Hlavním přínosem práce tedy byla implementace pěti algoritmů, z nichž čtyři byly zkonstruovány v rámci této práce.
Práce mně dala široký přehled v probírané problematice a motivaci problematiku inverzního geometrického ramene v geometrických algebrách rozšiřovat i do budoucna. V práci bych chtěl pokračovat tak, že bych tento problém a algoritmy nejen rozšířil do třídimenzionálního prostoru, ale také pomocí nich ovládal reálné robotické rameno, a tím nahradil matice rotací, které jsou výpočetně velice náročné. Chtěl bych zkusit se úplně vyhnout softwaru od výrobce robota, abych dosáhl co největšího výpočetního zjednodušení. V mé práci by se také dalo pokračovat přidáním možnosti zadávat přímo křivku místo série bodů v algoritmu pro

pohyb po lomené čáře. Dále by se dal nahradit algoritmus HVN za komplikovanější model robotického ramene.

Literatura

[1] Bruce, C. *Advanced Linear Algebra*. 3. vyd. CRC Press, 2010. ISBN 978-1-4398-2966-0.

[2] Calvet, R. G. *Treatise of Plane Geometry Through Geometric Algebra*. 1. vyd. [nakladatel není známý], 2007. ISBN 978-84-611-9149-9.

[3] Conformal Geometric Algebra, I. K. for the Industrial Robot IRB4400 Based on. *Radek Tichý* [online]. Springer, Cham, březen 2020. Dostupné z: https://link.springer.com/chapter/10.1007/978-3-030-43890-6_12.

[4] Doran, C. a Lasenby, A. *Geometric Algebra for Physicists*. 1. vyd. Cambridge University Press, 2003. ISBN 0-521-48022-1.

[5] Garling, D. J. H. *Clifford Algebras: An Introduction*. Cambridge University Press, 2011. ISBN 978-1-107-42219-3.

[6] Hestenes, D. *New Foundations for Classical Mechanics*. 2. vyd. Kluwer Academic Publishers, 2002. ISBN 0-7923-5514-8.

[7] Hestenes, D. *Geometric Algebra Computing in Engineering and Computer Science*. 1. vyd. Springer, 2010. ISBN 978-1-84996-107-3.

[8] Hildenbrand, D. *Foundations of Geometric Algebra Computing*. 1. vyd. Springer, 2013. ISBN 978-3-642-31793-4.

[9] Hildenbrand, D. *Introduction to Geometric Algebra Computing*. 1. vyd. Taylor & Francis Group, 2019. ISBN 978-1-4987-4838-4.

[10] Hrdina, J., Návrat, A. a Vašík, P. *Notes on Planar Inverse Kinematics Based on Geometric Algebra*. Springer, 2018. Dostupné z: https://doi.org/10.1007/s00006-018-0890-7.

[11] Leo Dorst, S. M. *Geometric Algebra for Computer Science*. 2. vyd. Elsevier Books, 2009. ISBN 978-0123749420.

[12] Machálek, L. *Korekce obrazových vad pomocí CGA*.

[13] Perwass, C. *Geometric Algebra with Applications in Engineering*. 1. vyd. Springer, 2009. ISBN 978-3-540-89067-6.

[14] Perwass, C. a Hildenbrand, D. *Aspects of Geometric Algebra in Euclidean, Projective and Conformal Space* [online]. 2004 [cit. 2022-4-04]. Dostupné z: http://www.gaalop.de/dhilden_data/CLUScripts/gatpdf.pdf.