Univerzita Palackého v Olomouci

Přírodovědecká fakulta

Katedra geoinformatiky

Martin DZÍBELA

IMPLEMENTACE ZÁKLADNÍCH FUNKCÍ FUZZY ARITMETIKY V C# S VYUŽITÍM V ARCGIS

Magisterská práce

Vedoucí práce: **doc. Mgr. Jiří DVORSKÝ, Ph.D.**

Olomouc 2016

Anotace

Diplomová práce se zabývá řešením problematiky fuzzy čísel, fuzzy aritmetiky a jejich využití při modelování neurčitosti a nejistoty v datech. Výstupem práce jsou C# knihovna FuzzyMath a nad ní postavené zásuvné moduly pro aplikaci ArcMap.

Klíčová slova

Fuzzy aritmetika, fuzzy čísla, po částech lineární fuzzy čísla, nejistota a neurčitost, GIS, ArcMap, .NET, C# knihovna

Počet stran práce: 51

Počet příloh: 1 (volná)

Anotation

The diploma thesis deals with fuzzy numbers, fuzzy arithmetic and its use in modeling uncertainty in measured data. Main outputs of the thesis are the C# library FuzzyMath and over it built add-in extensions of application ArcMap.

Keywords

Fuzzy arithmetic, fuzzy number, piecewise-linear fuzzy numbers, uncertainty, GIS, ArcMap, .NET, C# library

Number of pages: 51
Number of appendixes: 1

Prohlašuji, že

- jsem magisterskou magisterského studia oboru Geoinformatika vypracoval samostatně pod vedením doc. Mgr. Jiřího Dvorského, Ph.D. Všechny použité materiály a zdroje jsou citovány s ohledem na vědeckou etiku, autorská práva a zákony na ochranu duševního vlastnictví;
- jsem si vědom, že na moji diplomovou práci se plně vztahuje zákon č.121/2000 Sb. - autorský zákon, zejména § 35 – využití díla v rámci občanských a náboženských obřadů, v rámci školních představení a využití díla školního a § 60 – školní dílo;
- beru na vědomí, že Univerzita Palackého v Olomouci (dále UP Olomouc) má právo nevýdělečně, ke své vnitřní potřebě, diplomovou práci užívat (§ 35 odst. 3);
- souhlasím, aby jeden výtisk diplomové práce byl uložen v Knihovně UP k prezenčnímu nahlédnutí;
- souhlasím, že údaje o mé diplomové práci budou zveřejněny ve Studijním informačním systému UP;
- v případě zájmu UP Olomouc uzavřu licenční smlouvu s oprávněním užít výsledky a výstupy mé diplomové práce v rozsahu § 12 odst. 4 autorského zákona;
- použít výsledky a výstupy mé diplomové práce nebo poskytnout licenci k jejímu využití mohu jen se souhlasem UP Olomouc, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly UP Olomouc na vytvoření díla vynaloženy (až do jejich skutečné výše).

V Olomouci Martin Dzíbela

Rád bych tímto poděkoval doc. Jiřímu Dvorskému za vedení diplomové práce, RNDr. Cahikovi, Ph.D. BDP. SDP. za cenné rady a podmětné diskuze, rodičům za trpělivost, Míše a Dálovi za podporu a Janče za trpjelivost a korekturu. Děkuji.

[vložené zadání práce]

OBSAH

ÚVOD

Od uvedení teorie fuzzy množin, začátkem druhé poloviny minulého stolení, lze zaznamenat relativně krátké období, kdy tato matematická teorie začala nacházet reálné využití v průmyslových a jiných aplikacích. Fuzzy logika, která z teorie fuzzy množin přímo vychází, se uplatňuje v nejrůznějších řídících mechanismech, systémech kontroly kvality a zkrátka všude, kde konvekční matematika a logika naráží na své limity nebo přímo selhává [1].

Fuzzy aritmetika je jedna z matematických aplikací teorie fuzzy množin, která rozšiřuje klasickou aritmetiku o práci s fuzzy čísly, jakožto nepřesně zadanými, vágními, hodnotami. Fuzzy čísla totiž představují ideální model reprezentace nepřesných nebo neurčitých dat. Fuzzy číslo si lze představit jako libovolné reálné číslo spolu s jistou mírou neurčitosti či vágnosti, pramenící z jeho zadání (např. „asi čtyřicet“, „± 1492“, „kolem třetí“) nebo z přirozených chyb měřících přístrojů a samotného procesu sběru dat.

Zatímco fuzzy logika našla reálné praktické využití v podobě fuzzy regulátorů a na nich založených řídicích a rozhodovacích systémech, fuzzy aritmetika na něco podobného stále ještě čeká.

Jedním z důvodů, proč se tak zatím nestalo, může být už samotný přístup k chybám, respektive k nepřesnostem, ve vstupních datech. Obecně se na chyby a nepřesnosti v datech pohlíží jako na něco nežádoucího, špatného, něco, co do nich nepatří, a musí být proto odstraněno [2]. V tom lepším případě přijde na řadu konvenční statistika, v tom horším jsou tyto nepřesnosti zcela ignorovány. Tak jako tak, data, která následně vstupují do analýz a jejich dílčích výpočtů, jsou považována za přesná. Problém je, že přesná být nemohou, a to už jen proto, že jejich část byla nahrazena, případně zcela vypuštěna.

Postoj fuzzy aritmetiky k chybám či nepřesnostem, obecně k neurčitosti dat, je docela jiný. Neurčitost je zde chápána jako přirozená vlastnost většiny dat a informace o míře této neurčitosti je jejich důležitou součástí. Jinými slovy, data jsou a budou nepřesná, „pojďme s tím počítat“. Výstupem analýz a výpočtů využívajících fuzzy přístup a fuzzy aritmetiku bude vždy řešení, které zohledňuje všechny přípustné varianty pramenící z neurčitosti vstupních dat. To proto, že počáteční nejistota (tj. chyby v datech a obecně jejich neurčitost) na vstupu je propagována napříč všemi operacemi v rámci daného výpočtu.

Možnosti využití fuzzy přístupu k datům, jejich zpracovaní a analýzám v prostředí GIS byly již několikrát diskutovány. Ať již jde například o modelování a analýzy nad fuzzy povrchy [3, 4, 5, 6], síťové analýzy [7, 8], nebo třeba fuzzy dotazovaní

a extrakce neurčitých informací z dat [9, 10]. Je více než zřejmé, že fuzzy přístup k datům a jejich analýze dává, ve spojení s prostorovými daty, velký smysl a má zde obrovský potenciál.

Prvním krokem pro širší využití fuzzy přístupu, nejen v oblasti GIS, ale všude tam, kde má práce s nejistotou a neurčitostí v datech smysl (což zahrnuje téměř všechny přírodovědní obory), je poskytnout co možná nejvíce kvalitních nástrojů umožňujících tento přístup reálně aplikovat.

1 CÍLE PRÁCE

Cílem magisterské práce je vytvoření softwarové knihovny pro reprezentaci fuzzy čísel s možností základních aritmetických operací nad těmito čísly. Knihovna bude napsána na platformě .NET v programovacím jazyce C#. Následně bude vypracována případová studie, která ukáže možnosti využití v prostředí GIS.

V teoretické části práce bude zpracován stručný úvod do pojmů teorie fuzzy množin, fuzzy čísel a fuzzy aritmetiky, které jsou pro uchopení dané problematiky nezbytné. Dále budou popsány jednotlivé aspekty implementace fuzzy čísel a fuzzy aritmetiky, které musely být při návrhu knihovny zohledněny.

Stěžejním výstupem praktické části bude již zmiňovaná knihovna pro práci s fuzzy čísly. Smyslem implementace takové knihovny je vytvořit nástroj schopný modelovat míru nejistoty a neurčitosti v datech, kterou by zároveň umožnil propagovat do dalších výpočtů a analýz.

Spolu s knihovnou vzniknou i praktické příklady na využití fuzzy čísel a fuzzy aritmetiky v prostředí GIS. Konkrétně budou naprogramovány dvě ukázková rozšíření pro aplikaci ArcMap.

2 PRODUKTY A TECHNOLOGIE

V této kapitole budou představeny softwarové produkty a technologie, které byly při zpracování magisterské práce použity. Jmenovitě pak platforma .NET spolu s programovacím jazykem C# a geoinformační software ArcGIS for Desktop.

2.1 C# a platforma .NET

.NET (čteno „dotnet“) je označení pro technologickou platformu firmy Microsoft, která zastřešuje hned několik softwarových produktů a technologií [11]. Mezi ty patří například .NET Framework, nedávno představený .NET Core nebo třeba balíčkovací systém NuGet. Základním rysem .NET je multiplatformní vývojové a aplikační prostředí, které je nezávislé na volbě programovacího jazyka.

.NET Framework

.NET Framework je komplexní řešení realizující prostředí pro vývoj a běh aplikací na operačních systémech Windows. Tvoří jej rozsáhlá kolekce knihoven známá jako Framework Class Library (FCL) a virtuální strojového prostředí Common Language Runtime (CLR) [12]. Virtuální stroj CLR je vlastní implementace firmy Microsoft pro specifikaci Common Language Infrastructure (CLI), kterou původně vytvořila tatáž firma, jako základ platformy .NET.

Kód, který běží uvnitř virtuálního stroje, je označován jako „řízený“, naopak kód mimo jeho režii je „neřízený“. Neřízený kód je ve výsledku strojový kód, který je kompilován vždy na míru pro jednu konkrétní počítačovou architekturu. Jeho vykonávání proto bývá zpravidla velmi rychlé, avšak přináší i řadu nevýhod. Především pak náchylnost na chyby v běhu aplikace a nejrůznější bezpečnostní rizika. Ty vznikají zejména v důsledku špatné správy paměti a při práci s ukazateli (pointery) na konkrétní paměťové adresy.

Zdrojové kódy aplikací napsaných pro .NET se nekompilují do nativního strojového kódu, ale jsou nejprve přeloženy do jakéhosi mezi-jazyka. Ten je známý jako Common Intermediate Language (CIL) a je nezávislý jak na operačním systému, tak i na architektuře daného počítače [12]. Následné spuštění a běh kódu obstarává virtuální stroj, třeba právě CLR. Způsob jakým má tento stroj kód jazyka CIL provádět a obsluhovat, včetně vlastností samotného CIL, nařizuje specifikace CLI.

C#

Programovací jazyk C# (čteno „C sharp“) je víceúčelový objektově orientovaný programovací jazyk, který byl vytvořen firmou Microsoft, aby splňoval podmínky specifikace CLI a stal se referenčním jazykem pro platformu .NET [12]. V současnosti je jazyk C# ve verzi 6.0, která byla vydána v červenci 2015, společně s .NET Framework ve verzi 4.6 a vývojovým prostředím Visual Studio 2015.

Syntaxí se C# velmi podobá jazykům jako C, C++ a nebo Java; viz následující ukázky kódu, včetně obligátní „Hello, World!“ aplikace (Zdrojový kód 1).

Zdrojový kód 1

```
using System;

class Program
{
    static void Main()
    {
        Console.WriteLine("Hello, World!");
    }
}
```

C# je staticky typovaný jazyk s možností implicitní kontroly typu za běhu. Od verze 2.0 byly do jazyka zavedeny generické typy, tzn. parametrizované typy, které jsou instancované až za běhu aplikace. Díky tomu je možné vytvářet vysoce flexibilní, a zároveň typově bezpečné komponenty.

Velice pohodlně řeší jazyk přístup k členským proměnným objektů v podobě třídních vlastností objektu. Programátor se tak může elegantně vyhnout psaní přístupových metod a metod pro jednoduchou změnu nebo nastavení členských proměnných, tzv. getterů a setterů; viz Zdrojový kód 2.

Zdrojový kód 2

```
class Person
{
    private int age = 0;

    public int Age
    {
        protected set
        {
            if (value < 0)
            {
                throw new Exception("Invalid age");
            }
            age = value;
        }
```

```
        get { return age; }
    }
}
```

Ve verzi jazyka C# 3.0 byl společně s .NET Framework 3.5 uveden nový integrovaný dotazovací jazyk Language Integrated Query (LINQ). LINQ přináší intuitivní syntaxi pro přístup a manipulaci s daty, bez ohledu na jejich původ; viz Zdrojový kód 3.

Zdrojový kód 3

```
var numbers = new List<int>() { 1, 2, 5, 2, 11, 3, 9, 8 };

var even = form num in numbers
        where num % 2 = 0
        orderby num
        select num;
```

Předchozí ukázka s využitím metod rozhraní `IEnumerable` a lambda výrazů:

```
var numbers = new List<int>() { 1, 2, 5, 2, 11, 3, 9, 8 };

var even = numbers.Where(x => x % 2 = 0).OrderBy(num => num);
```

2.2 ArcGIS

ArcGIS for Desktop je komplexní desktopový GIS, patřící do produktové řady komerční platformy ArcGIS firmy ESRI. ArcGIS for Desktop tvoří balíček několika vzájemně provázaných aplikací, aplikačních rozšíření a nadstaveb.

ArcGIS for Desktop je k dispozici ve třech licenčních úrovních: Basic, Standard a Advanced [13]. Ty se vzájemně odlišují na základě poskytované funkcionality. Nejnižší licence, Basic, slouží především k prezentaci a analýze geografických dat, případě k jejich jednoduché správě a editaci. Licenční úroveň Standard rozšiřuje možnosti úrovně Basic o pokročilé způsoby správy a kontroly dat. Naplno využívá potenciálu geodatabází pro práci s vektory a jejich topologií. Umožňuje vytvářet pokročilá pravidla kartografické reprezentace, včetně tvorby anotací vázaných na konkrétní geografické prvky. Profesionální desktopové GIS bez kompromisů pak představuje ArcGIS for Desktop Advanced.

ArcMap je centrální aplikací ArcGIS for Desktop. Poskytuje uživatelské rozhraní nad rozsáhlou kolekcí analytických, editačních a publikačních nástrojů. S nejnovější verzí platformy ArcGIS 10.4 byla kolekce aplikací ArcGIS for Desktop rozšířena o zcela nový desktopový GIS – ArcGIS Pro, běžící na moderní 64 bitové

architektuře s podporou paralyzace výpočtů v případě víceprocesorových systémů [13]. Zaměřením si jsou obě aplikace poměrně blízké, a ačkoliv ArcGIS Pro v současné době nedokáže plně pokrýt funkčnost ArcMap a ESRI podobné domněnky vyvrací, je docela možné, že jej v budoucnu zcela nahradí.

Pro zpracování případové studie v rámci praktické části této diplomové práce nebyla využita nejnovější verze ArcGIS for Desktop 10.4, ale o dva produkční cykly nižší verze 10.2, a to v licenční úrovni ArcGIS for Desktop Advanced.

Vývoj s ArcGIS for Desktop

Funkčnost aplikací ArcGIS for Desktop lze relativně snadno rozšířit a přizpůsobit pomocí uživatelských add-in modulů. ArcGIS for Desktop Add-in představuje model pro vývoj a začleňování uživatelských rozšíření do desktopových aplikací na platformě ArcGIS 10 a vyšší.

Aplikace ArcGIS for Desktop (vyjma ArcGIS Pro) jsou postaveny nad rozsáhlou knihovnou softwarových komponent – ArcObjects [14]. Knihovna ArcObjects je napsaná v jazyce C++ a využívá technologického standardu Component Object Model (COM), který zajišťuje nezávislost jak na produkční, tak i vývojové platformě. Jednotlivé komponenty a funkce knihovny ArcObjects jsou dostupné prostřednictvím několika aplikačních programovacích rozhraní: .NET (C#, VB.NET, VC++) API, Java API a C++ API.

Platfoma .NET, respektive .NET Framework, nabízí mechanismy, jak s COM objekty obousměrně komunikovat [11]. Základ pro tuto spolupráci představuje Runtime Callable Wrapper (RCW), který pro spravovaný kód běhového prostředí CLR zpřístupňuje nespravovaný kód v podobě COM objektů. RCW v tomto procesu vystupuje jako jakýsi proxy objekt poskytující rozhraní konkrétní COM komponenty. Komunikaci opačným směrem, tedy kdy COM, jakožto klient, volá spravovaný kód .NET komponenty, pak zajišťuje COM Callable Wrapper (CCW).

Aby spolupráce .NET-COM mohla bez problému fungovat, musí mít běhové prostředí .NET (CLR) k dispozici popis typové knihovny zpřístupňovaných COM komponent. Ideální řešení pak představuje Primary Interop Assembly (PIA), oficiální sestavení dodané a podepsané autorem COM knihovny.

ArcObjects SDK

ArcObjects SDK for the Microsoft .NET Framework (dále jen jako ArcObjects SKD) je sada vývojářských nástrojů pro umožnění vývoje nad knihovnou ArcObjects

v prostředí platformy .NET [14]. Balíček ArcObjects SDK je dodáván spolu s vybranými produkty ArcGIS (např. ArcGIS for Desktop, ArcGIS for Server nebo ArcGIS Engine). Mimo rozsáhlé a nepřehledné dokumentace obsahuje i kompletní kolekci PIA sestavení, kterou lze po instalaci SDK nalézt v globálním repozitáři Global Assembly Cache (GAC).

Součástí ArcObjects SDK je také řada užitečných nástrojů a rozšíření pro vývojové prostředí Visual Studio, jako jsou snippety – malé útržky znovupoužitelného kódu, nebo předpřipravené šablony vybraných typů ArcGIS projektů.

3 SOUČASNÝ STAV ŘEŠENÉ PROBLEMATIKY

V úvodu této diplomové práce byla zmíněna jistá neochota a nedostatek vůle pro reálnou aplikaci fuzzy aritmetiky a obecně modelovávání nejistoty či nepřesnosti v datech pomocí fuzzy čísel. Jako jedna z možných příčin proč tomu tak je, bylo uvedeno konvenční pojetí neurčitosti a vágnosti v datech, jako chyby, které je třeba se zbavit.

Dalším a dost možná i tím nejpodstatnějším důvodem, který brání rozšíření fuzzy přístupu k datům, jejich zpracování a analýze do reálných aplikací, je nedostatek základních nástrojů pro podporu fuzzy aritmetiky a nástrojů pro práci s fuzzy čísly obecně [15].

Pokud již nějaké takové nástroje existují, mají většinou jen charakter vědeckých nástrojů, orientovaných na velice specifické úlohy, aniž by poskytovaly možnosti pro řešení obecných problémů. Příkladem může být balíček Fuzzy Logic Toolbox komerčního matematického softwaru MATLAB od MathWorks; viz oficiální webové stránky [16]. Případně další komerční řešení v podobě produktu Fuzzy Logic pro Wolfram Mathematica; viz online dokumentace [17]. Již z jejich názvů lze vytušit, že jde především o nástroje pro práci s fuzzy logikou a na jejich principech založených systémech, než pro práci s fuzzy čísly. V obou případech je fuzzy aritmetika implementována takovým způsobem, že prakticky vylučuje jakékoliv smysluplné využití v reálných výpočetních úlohách a její účel je zde především propagační.

Balíček FuzzyNumbers

V současné době asi jediný použitelný nástroj pro reprezentaci fuzzy čísel, schopný základních aritmetických operací s těmito čísly, je R[1] balíček *FuzzyNumbers* [18]. Balíček byl vydán v roce 2015 pod svobodnou licencí LGPL (GNU Lesser General Public License) a celý projekt je dostupný ve veřejném Git repozitáři na serverech populární webové služby pro hostování open-source softwaru – GitHub. Autorem balíčku je Marek Gągolewski, jako spoluautor je uveden Jan Caha.

Podle oficiálních stránek balíčku FuzzyNumbers je jeho cílem poskytnout nástroje pro práci s fuzzy čísly a fuzzy aritmetikou, a to jak pro praktické, tak i výzkumné účely.

[1] R je interpretovaný programovací jazyk a open-source prostředí pro statistickou analýzu dat.

Balíček FuzzyNumbers, v aktuální podobě, nabízí funkčnost pro

- reprezentaci fuzzy čísel, zadaných svými α-řezy, jako po částech lineární fuzzy čísla, případně dvojicí nespojitých funkcí;
- defuzzifikaci po částech lineárních fuzzy čísel;
- grafickou vizualizaci fuzzy čísel;
- základní aritmetické operace nad fuzzy čísly.

Problém a omezení balíčku FuzzyNumbers může představovat jeho závislost na prostředí R. Případně pak samotný jazyk R, který, jakožto interpretovaný programovací jazyk, není příliš vhodný pro zpracovávání většího množství dat; např. fuzzifikovaného rastru digitálního modelu terénu.

Cílem této diplomové práce je poskytnout vlastní univerzální řešení pro podporu fuzzy čísel a fuzzy aritmetiky v podobě softwarové knihovny pro platformu .NET. Tato knihovna by pak měla sloužit jako základ k vývoji konkrétních aplikací a nástrojů pracujících s fuzzy čísly a fuzzy aritmetikou.

4 TEORETICKÉ PODKLADY

Náplní této kapitoly je stručný úvod do pojmů teorie fuzzy množin, fuzzy čísel a fuzzy aritmetiky.

4.1 Základní pojmy teorie fuzzy množin

Klasickou (ostrou) množinu je možné zadat definováním její tzv. *charakteristické funkce*, která určuje příslušnost prvku $x \in U$ do množiny $A \subseteq U$. Pro funkci platí následující vztah

$$\chi_A(x) = \begin{cases} 1 & \text{pokud } x \in A \\ 0 & \text{jinak} \end{cases}, \qquad (1)$$

ze kterého vyplývá, že každý prvek z universa U do množiny A buď patří, anebo nepatří. Oproti tomu fuzzy množina, tak jak ji definoval Zadeh [19], může obsahovat prvky na různých stupních příslušnosti.

Jako příklad lze zadat množinu „státy střední Evropy“. Jaké státy do takové množiny zařadit? Pro začátek to mohou být třeba země Visegrádské čtyřky, tedy Česko, Slovensko, Polsko a Maďarsko. Co Alpské země? – Slovinsko, Rakousko, Německo, Švýcarsko, Lichtenštejnsko. Jsou i ony součástí této množiny? A co pobaltské země? Problém klasických ostrých množin je, že umí rozlišovat pouze mezi dvěma extrémy; stát do střední Evropy patří, anebo nepatří. Fuzzy množiny takový problém netrápí. Každý prvek, který je součástí nějaké fuzzy množiny, je její součástí na určité hladině (stupni) příslušnosti vyjadřující míru jistoty v tvrzení, že do této množiny skutečně patří. V konkrétním případě fuzzy množiny „státy střední Evropy“ lze takto vyjádřit právě onu míru nejistoty, zda daná země do této množiny patří, či nepatří.

Stupeň příslušnosti

Zobecněním charakteristické funkce (1), tak aby příslušnost prvků $x \in U$ k množině $A \subseteq U$ nabývala spojitých hodnot od 0 do 1, je tzv. *funkce příslušnosti*

$$\mu_A : U \to \langle 0, 1 \rangle. \qquad (2)$$

Hodnota $\mu_A(x)$ se nazývá *stupeň příslušnosti* a indikuje, jak moc je prvek x součástí fuzzy množiny A. Pro hodnotu $\mu_A(x) = 1$ platí, že x je součástí fuzzy množiny A zcela úplně, naopak pro hodnotu 0 platí, že prvek x do dané fuzzy množiny vůbec nepatří. Zbylé hodnoty značí pouze částečnou příslušnost, kdy prvek x je prvkem

fuzzy množiny A na stupni příslušnosti $0 < \mu_A(x) < 1$, ale zároveň je také prvkem jejího doplňku, fuzzy množiny A^C, na stupni příslušnosti $\mu_{A^C}(x) = 1 - \mu_A(x)$.

α-řez

Jako *α-řez* fuzzy množiny $A \subseteq U$ na stupni příslušnosti α je označována ostrá množina

$$A_\alpha = \{x \in U \mid \mu_A(x) \geq \alpha\}, \tag{3}$$

kde $\alpha \in \langle 0, 1 \rangle$.

Jádro fuzzy množiny A je ostrá množina $KerA$, shodná s α-řezem na stupni příslušnosti $\alpha = 1$. *Podstava* fuzzy množiny A je ostrá množina $SuppA$, obsahující všechny x, pro které platí $\mu_A(x) > 0$.

Stejně jako je možné klasickou ostrou množinu zadat výčtem svých prvků, lze i fuzzy množinu definovat výčtem α-řezů, jako $A = \{A_{\alpha_1}, A_{\alpha_2}, \dots, A_{\alpha_n}\}$, kde pro každý α-řez $A_{\alpha_{k+1}}$ na stupni příslušnosti $\alpha_{k+1} > \alpha_k$ platí[2], že je podmnožinou A_{α_k} [20].

Takováto reprezentace fuzzy čísel může být leckdy výhodnější, zejména při volbě vhodné aproximace funkce příslušnosti, která vede ke snížení konečného počtu α-řezů [15].

Vlastnosti fuzzy množin

Caha [3] zmiňuje dvě základní vlastnosti fuzzy množin – konvexnost a výšku. Fuzzy množina je *konvexní* právě tehdy, jsou-li konvexní i všechny ostré množiny představující její α-řezy; v opačném případě je fuzzy množina *nekonvexní* [21].

Výškou fuzzy množiny $A \subseteq U$ se pak rozumí maximální hodnota stupně příslušnosti prvku $x \in U$ k fuzzy množině A

$$\mathrm{hgt}(A) = \sup \mu_A(x). \tag{4}$$

Je-li výška $\mathrm{hgt}(A)$ rovna 1, fuzzy množina A se nazývá *normální*; v opačné případě *nenormální*.

---

[2] Jedná se o přirozenou vlastnost α-řezů vyplívající z definice (3).

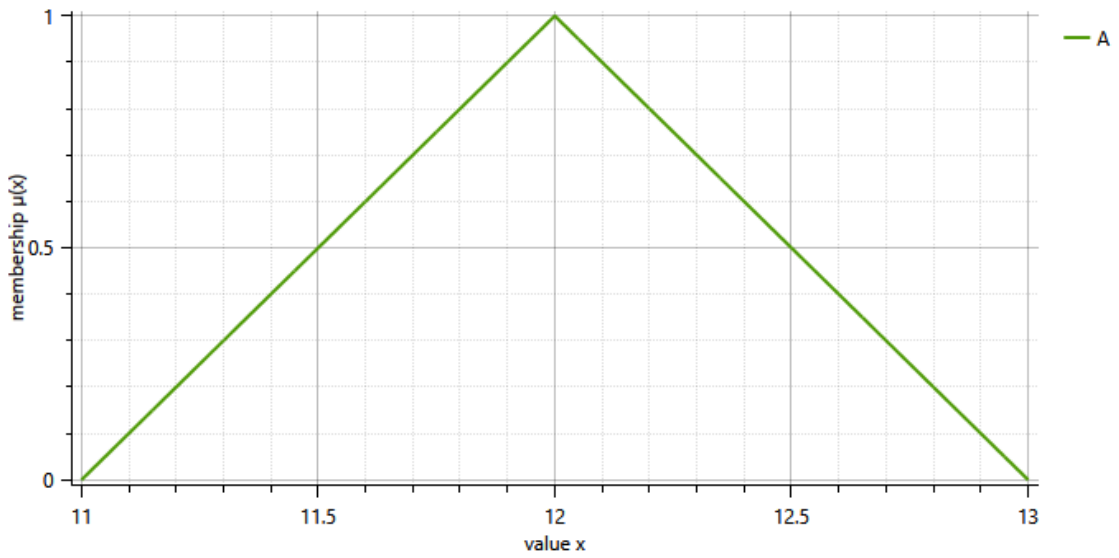
4.2 Fuzzy čísla

Fuzzy čísla jsou speciálním případem fuzzy množin definovaných nad množinou reálných čísel $\mathbb{R}$. Fuzzy množina $A \subseteq \mathbb{R}$ je fuzzy číslem jestliže splňuje následující vlastnosti [15, 21, 22]:

a. A je normální fuzzy množina,
b. A je konvexní,
c. funkce příslušnosti (2) je alespoň po částech spojitá.

Význam fuzzy čísel spočívá především v jejich schopnosti modelovat nejistotu a neurčitost, jako přirozenou součást nějaké informace. Jako příklad fuzzy čísla lze uvést celkem běžný časový údaj „kolem poledne“. Slovo „poledne“ označuje 12 hodin místního času a příslovce „kolem“ je právě ona míra nejisty. Obrázek 1 zobrazuje možnou podobu fuzzy čísla 12, jakožto reprezentaci času „kolem poledne“, neboli „asi 12 hodin“.

Obrázek 1 – fuzzy číslo „asi 12“

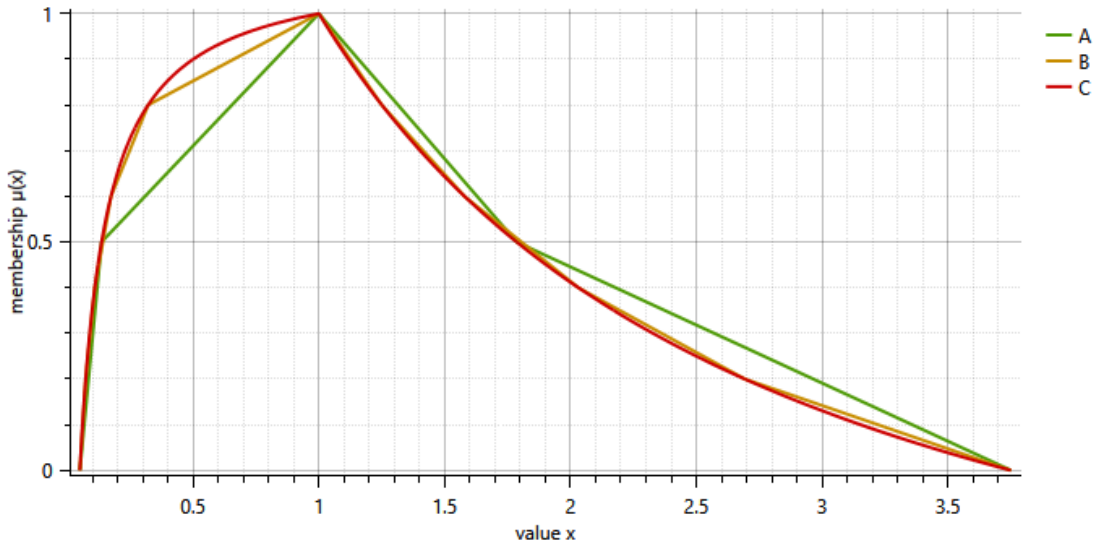


Po částech lineární fuzzy čísla

Po částech lineární fuzzy čísla jsou speciálním případem fuzzy čísel, definovaných krajními body svých α-řezů [23]. Každé takové číslo je možné popsat $2n + 4$ body. Pro zadání α-řezu, jakožto intervalu na množně reálných čísel, jsou nutné dva body; dolní a horní mez intervalu. Počet těchto dvojic bodů vyjadřuje právě n. Dále platí, že každá fuzzy množina má svoji podstavu a zároveň každé fuzzy číslo je normální, má tedy jádro (α-řez na stupni příslušnosti 1).

Pro zadání po částech lineárního fuzzy čísla jsou proto nutné minimálně čtyři reálná čísla, představující hraniční body podstavy a jádra. Zbylých $2n$ bodů pak slouží k případnému zpřesnění popisu průběhu funkce příslušnosti daného fuzzy čísla.

Obrázek 2 – po částech lineární fuzzy čísla A, B, C



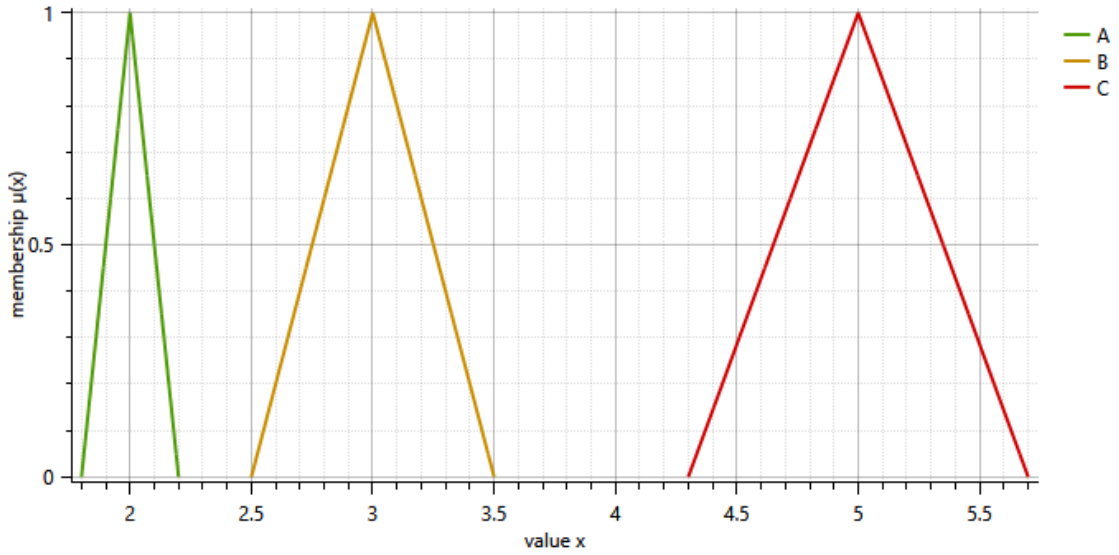
Obrázek 2 zobrazuje závislost počtu α-řezů na výslednou přesnost po částech lineárních fuzzy čísel. Fuzzy číslo *A* bylo zadáno pouze pomocí 3 α-řezů, fuzzy číslo *B* již pomocí 6 a fuzzy číslo *C* pomocí 101 α-řezů. Je jasně patrné, že čím vyšší počet α-řezů (u fuzzy čísla s nelineárním průběhem funkce příslušnosti), tím vyšší je přesnost jeho zobrazení jakožto po částech lineárního fuzzy čísla. Nicméně je dobré si zároveň uvědomit, že příliš vysoký počet α-řezů má negativní vliv na výkon a paměť prováděných operací nad těmito čísly.

4.3 Fuzzy aritmetika

Fuzzy aritmetika rozšiřuje klasikou aritmetiku o výpočty s fuzzy čísly [22]. Fuzzy čísla zde představují nepřesné nebo neurčité hodnoty a fuzzy aritmetika nabízí způsob, jak s těmito hodnotami pracovat. Jako příklad lze uvést nejasně zadaná čísla „asi dva“ a „zhruba tři“. Výsledkem jejich součtu by měla být hodnota, kterou lze popsat jako „asi zhruba pět“. Důležité je, že při operacích nad fuzzy čísly by nemělo docházet ke ztrátě informace o nepřesnosti nebo neurčitosti dané hodnoty, kterou tato čísla reprezentují. Proto výsledkem předchozího součtu není „asi pět“, ani „zhruba pět“. Do výsledku je promítnuta nejistota z obou zadaných

čísel, vyjádřená slovy „asi“ a „zhruba“; viz Obrázek 3, kde A ~ „asi dva“, B ~ „zhruba tři“ a C je pak jejich součet.

Obrázek 3 – fuzzy aritmetika: C = A + B



Princip rozšíření

Jednou ze základních myšlenek teorie fuzzy množin, publikovaných v [19], je princip rozšíření. Fuzzy množinu lze formálně popsat jako zobrazení prvků klasické ostré množiny X do intervalu $\alpha \in \langle 0, 1 \rangle$. Podle principu rozšíření pak pro každou funkci $f: X \to Y$ musí existovat odpovídající funkce $\tilde{f}: \alpha^X \to \alpha^Y$. Princip rozšíření tedy dovoluje rozšířit libovolnou operaci klasické aritmetiky na její fuzzy alternativu.

Implementace fuzzy aritmetiky, jako přímé aplikace principu rozšíření, naráží na obtížně řešitelný problém spojený s potřebou nalezení globálních extrémů, respektive globálních maxim a minim funkcí, případně některé další problémy [15, 21, 22]. Proto se v praxi daleko častěji využívají alternativní přístupy [3]; např. fuzzy aritmetika pomocí α-řezů.

Fuzzy aritmetika pomocí α-řezů

Každé fuzzy číslo lze více či méně přesně popsat jako uspořádanou konečnou množinu intervalů, představujících jeho α-řezy. Libovolná (aritmetická) operace nad fuzzy čísly tak může být vyjádřena jako tatáž operace nad korespondujícími intervaly z těchto množin.

Pro intervalovou aritmetiku platí, že výsledkem jakékoliv operace je pokaždé nejširší možný interval všech přípustných řešení [24]. Formálně lze intervalovou aritmetiku definovat jako

$$A \oplus B = \{ x \mid \exists y \in A \, \exists z \in B \; x = y \oplus z \}. \qquad (5)$$

Problém závislostí

Bohužel ani intervalová aritmetika není tak úplně bezproblémová. Asi největší problém intervalové aritmetiky představuje tzv. *dependency problem* (problém závislostí) [25]. Jako příklad, kde a jak se tento problém projevuje, lze uvést interval $x \in \langle -2, 2 \rangle$ a složenou operaci nad tímto intervalem $x^2 - x$. Přirozený postup $\langle -2, 2 \rangle^2 + \langle -2, 2 \rangle = \langle 0, 4 \rangle + \langle -2, 2 \rangle = \langle -2, 6 \rangle$ nepovede ke správnému výsledku, protože neřeší příklad jako celek, ale odděleně, tedy jako $x^2 - y$, kde $x, y \in \langle -2, 2 \rangle$.

Neexistuje žádný univerzální postup, jak tento problém řešit. Jedinou možností je využít znalost dané problematiky a tam, kde je to možné, upravit zadání tak, aby tento problém nemohl nastat. V předchozím příkladu lze zadání $x^2 - x$ upravit na ekvivalentní tvar $(x - 1/2)^2 - 1/4$. Správné řešení předchozího příkladu by pak vypadalo jako $(\langle -2, 2 \rangle - 1/2)^2 - 1/4 = \langle -5/2 \, , 3/2 \rangle^2 - 1/4 = \langle -1/4 \, , \; 6 \rangle$.

5 POSTUP IMPLEMENTACE FUZZY ARITMETIKY V C#

Hlavním cílem diplomové práce bylo navrhnout a vytvořit knihovnu pro platformu .NET v programovacím jazyce C#, která by řešila problematiku fuzzy čísel a fuzzy aritmetiky. V této kapitole bude detailně popsán postup návrhu a následné implementace jednotlivých částí této knihovny.

5.1 Implementace fuzzy čísla

Při zpracovávání teoretických podkladů pro návrh fuzzy čísla a fuzzy aritmetiky vyplynulo, že zdaleka nejvýhodnějším přístupem bude implementovat fuzzy číslo jako po částech lineární. Po částech lineární fuzzy číslo lze popsat jako konečnou množinu reálných intervalů, představujících jeho α-řezy. Prvním krokem tak logicky byla implementace samotného intervalu.

Interval jako reprezentace α-řezu

Interval je množina reálných čísel, která je vymezena horní a dolní mezí. Podle toho, zda je interval otevřený, otevřený zleva nebo otevřený zprava, lze rozlišit, jestli jsou meze součástí intervalu. Pro potřeby reprezentace α-řezu fuzzy čísla jsou všechny intervaly uzavřené; meze jsou tedy součástí daného intervalu.

Implementace samotného intervalu byla poměrně přímočará. Nicméně i v případě takto primitivní datové struktury bylo třeba pečlivě namyslet a vyřešit několik technických otázek. Tou nejpodstatnější byla volba vhodného datového typu pro reprezentaci reálného čísla. Platforma .NET disponuje hned třemi číselnými datovými typy, které jsou pro tyto účely více či méně vhodné. `Double` a `float` jsou datové typy pro binární reprezentaci číselných hodnot s pohyblivou řádovou (desetinou) čárkou. Rozdíl mezi nimi je pouze v počtech bitů a s tím spojeným rozsahem a přesností v počtu desetinných míst. `Decimal` je další datový typ určený k aritmetice s pohyblivou řádovou čárkou. Oproti předchozím dvěma typům používá k tomuto účelu desítkovou soustavu, ne binární. Díky tomu nedochází k tak častým aproximacím hodnot za řádovou čárkou, jako u typů `double` nebo `float`. Nicméně, dojde-li na aritmetické operace, je binární reprezentace čísel mnohem efektivnější, a proto bylo rozhodnuto, že výchozím datovým typem pro implementaci intervalu bude `double`.

Problém binární reprezentace čísel je neschopnost přesné reprezentace hodnot za pohyblivou řádovou čárkou. Tyto hodnoty musí být aproximovány a často tak dochází k nečekaným a nechtěným situacím; viz Zdrojový kód 4.

Zdrojový kód 4

```
double value = .1;

double x = value * 10;
double y = 0;

for (var i = 1; i <= 10; i++)
{
    y += value;
}

if (x == y)
{
    Console.WriteLine("{0:R} == {1:R}", x, y);
}
else
{
    Console.WriteLine("{0:R} != {1:R}", x, y);
}
```

Nečekaně na výstupu konzole nebude „1 == 1“, ale „1 != 0,99999999999999989“, jakožto důsledek již zmiňované aproximace. Řešením výše popsaného problému může být přidání určité míry tolerance, která bude zohledňována při každé kontrole rovnosti mezi dvěma hodnotami; tak jako to ukazuje příklad níže.

Zdrojový kód 5

```
static bool AreSame(double x, double y, double epsilon)
{
    return Math.Abs(a - b) <= epsilon;
}
```

Zdrojový kód 5 definuje jednoduchou funkci `AreSame()`, která umí rozhodnout, zda si jsou, v rámci dané tolerance, čísla *x* a *y* rovna. Výstupem funkce je *true* (pravda), pokud je absolutní hodnota rozdílu obou čísel menší nebo rovna hodnotě tolerance epsilon; v opačném případě vrátí funkce *false* (nepravda).

Otázkou nadále zůstává ideální hodnota míry tolerance epsilon. Odpověď na ni však není ani zdaleka jednoznačná a vždy záleží na konkrétní řešené situaci. Nicméně vzhledem k faktu, že reálný počet desetinných míst, které může datový typ double obsáhnout, je asi 15, pohybuje se smysluplný rozsah pro zadání hodnoty epsilon v rozmezí od 10^{-1} do 10^{-15}.

Implementace intervalu

Před samotnou implementací intervalu bylo zvažováno několik možností, jak do jeho struktury zavést výše popsanou míru tolerance pro rovnost dvou hodnot

datového typu `double`. Jednou z nich bylo definovat hodnotu tolerance epsilon jako konstantu. Vzhledem k individuálním nárokům na hodnotu takové konstanty, není toto řešení příliš flexibilní, a bylo proto odmítnuto. Epsilon musí být nastavitelné dle potřeb konkrétních úloh. Globální parametr pro hodnotu epsilon rovněž nepřicházel v úvahu – fuzzy číslo a tedy i interval, jakožto jeho α-řez, musí být imutabilní, tzn. neměnný, stálý. Je nepřijatelné, aby ho zvenčí ovlivňoval nějaký globální parametr. Ze všech ostatních řešení se nakonec jako to nepraktičtější ukázalo prosté přidání epsilonu mezi parametry pro vytvoření instance intervalu.

Interval

Zdrojový kód 6 ukazuje základ pro implementaci intervalu. Konstruktor struktury `Interval` má dva povinné argumenty definující horní – *a* – a dolní – *b* – mez intervalu. Třetí argument, *epsilon*, je volitelný a představuje již zmiňovanou míru tolerance, do které lze dvě hodnoty číselného typu `double` považovat za sobě rovné. Horní a dolní hranice, stejně jako hodnota epsilon, jsou zpřístupněné jako veřejné vlastnosti instance intervalu.

Zdrojový kód 6

```
public struct Interval
{

    public Interval(double a, double b, double epsilon = 0)
    {
        this.epsilon = epsilon;

        if (Math.Abs(a - b) <= epsilon)
        {
            this.a = this.b = a;
        }
        else if (a < b)
        {
            this.a = a;
            this.b = b;
        }
        else
        {
            throw new ArgumentException();
        }
    }

    public double Epsilon
    {
        get { return epsilon; }
    }

    public double A
    {
        get { return a; }
```

```
    }

    public double B
    {
        get { return b; }
    }

    private double a, b, epsilon;

}
```

Součástí úplné implementace struktury intervalu je i několik užitečných metod. Tou nejdůležitější je pak přetížená metoda `Contains()`. Z názvu metody lze poměrně snadno vytušit, k čemu slouží. Metoda jednoduše vrací pravdivostní hodnotu (`bool`), zda instance intervalu obsahuje konkrétní prvek, případně nějaký další interval, specifikovaný v argumentu metody.

Implementace fuzzy čísla

Jakmile byl vytvořen α-řez, respektive interval, který jej reprezentuje, mohlo být přistoupeno k implementaci po částech lineárního fuzzy čísla. Po částech lineární fuzzy číslo (dále jen fuzzy číslo) lze popsat jako množinu α-řezů na odpovídajících hladinách stupňů příslušnosti. Za předpokladu rovnoměrného zastoupení α-řezů v rámci daného fuzzy čísla, lze z jejich počtu hladiny příslušnosti poměrně snadno odvodit. Pro takové fuzzy číslo by pak platilo, že má n α-řezů na hladinách příslušnosti $\alpha_i = \frac{i-1}{n-1}$.

FuzzyNumber

Zdrojový kód 7 ukazuje základní kostru třídy pro reprezentaci fuzzy čísla. Konstruktor očekává kolekci intervalů jakožto α-řezů, nad kterými bude dané fuzzy číslo definováno. První v pořadí z kolekce intervalů je považován za α-řez na stupni příslušnosti $\alpha_1 = 0/n$, druhý za α-řez na stupni příslušnosti $\alpha_2 = 1/n$, třetí v pořadí za α-řez na stupni příslušnosti $\alpha_3 = 2/n$, atd. až k poslednímu intervalu, α-řezu na hladině příslušnosti $\alpha_n = n/n$. V případě, že kolekci tvoří pouze jeden jediný interval, předpokládá se, že podstava i jádro fuzzy čísla jsou shodné a odpovídají právě tomuto intervalu. Prázdná kolekce vyvolá výjimku. Zároveň musí platit, že každý interval $A_{\alpha_{k+1}}$ z uspořádané kolekce vstupních intervalů je podmnožinou intervalu A_{α_k} téže kolekce. Pokud tomu tak není, konstruktor fuzzy čísla vyvolá výjimku.

Zdrojový kód 7

```
public class FuzzyNumber
{

    public FuzzyNumber(IEnumerable<Interval> intervals)
    {
        var alphaCuts = new List<Interval>();

        var maxA = double.NegativeInfinity;
        var minB = double.PositiveInfinity;

        foreach (var interval in intervals)
        {
            maxA = Math.Max(interval.A, maxA);
            minB = Math.Min(interval.B, minB);

            if (interval.Contains(maxA) && interval.Contains(minB))
            {
                alphaCuts.Add(interval);
            }
            else
            {
                throw new ArgumentException();
            }
        }

        if (alphaCuts.Count > 0)
        {
            AlphaCuts = new ReadOnlyCollection<Interval>(alphaCuts);
        }
        else
        {
            throw new ArgumentException();
        }
    }

    public ReadOnlyCollection<Interval> AlphaCuts;

}
```

Podoba fuzzy čísla je tedy definována kolekcí intervalů, α-řezů, v konstruktoru jeho třídy. Jakmile je jednou fuzzy číslo inicializované, není už možné do něj jakkoliv zasahovat; tzn. instance fuzzy čísla je imutabilní.

Jednotlivé α-řezy tvořící fuzzy číslo, jsou zpřístupněné přes veřejnou vlastnost `AlphaCuts` dané instance, a to jako uspořádaná kolekce objektů typu `Interval`. Pro usnadnění práce s fuzzy číslem je první a poslední prvek této kolekce zpřístupněn veřejnou vlastností `Support`, respektive `Kernel`. Pro získání α-řezu na konkrétním stupni příslušnosti slouží metoda `GetAlphaCut()` se stupněm příslušnosti jako argumentem (`double` od 0 do 1). Metoda vrací α-řez jako instanci intervalu.

Další důležitou metodou, kterou třída `FuzzyNumber` disponuje, je metoda funkce příslušnosti `GetMembership()`. Ta, jak už její název napovídá, vrací hodnotu stupně příslušnosti (`double` od 0 do 1) daného prvku, coby argumentu (`double`) metody, k instanci fuzzy čísla.

FuzzyNumberFactory

Vytvářet fuzzy číslo jen z kolekce α-řezů by bylo značně nepraktické a nepohodlné. Nejběžnější způsob pro zadání fuzzy čísla je pomocí posloupnosti tří až čtyř čísel, které představují mezní body jeho podstavy a jádra: $Supp_1$, Ker_1, Ker_2, $Supp_1$; pro posloupnost tří čísel pak platí, že $Ker_1 = Ker_2$. Proto byla vytvořena třída `FuzzyNumberFactory`, viz Zdrojový kód 8, umožňující vytvářet fuzzy čísla právě z takovýchto posloupností.

Zdrojový kód 8

```
public class FuzzyNumberFactory
{

    public FuzzyNumberFactory(int num = 11, double epsilon = 1E-6)
    {
        NumberOfAlphaCuts = num > 1 ? num: 2;
        Epsilon = epsilon;
    }

    public FuzzyNumber CreateTrapezoidal(double a, double b,
                                         double c, double d)
    {
        var intervals = new Interval[NumberOfAlphaCuts];

        for (var i = 0; i < NumberOfAlphaCuts; i++)
        {
            intervals[i] = new Interval(
                    a + (b - a) * i / (NumberOfAlphaCuts - 1),
                    d - (d - c) * i / (NumberOfAlphaCuts - 1),
                    Epsilon
                );
        }

        return new FuzzyNumber(intervals);
    }

    public FuzzyNumber CreateTriangular(double a, double b, double c)
    {
        return CreateTrapezoidal(a, b, b, c);
    }

    public FuzzyNumber CreateCrisp(double a)
    {
        return CreateTrapezoidal(a, a, a, a);
    }
```

```
    public int NumberOfAlphaCuts { get; private set; }
    public double Epsilon { get; private set; }
}
```

Pro vytvoření fuzzy čísla výše uvedeným způsobem je nutné znát počet jeho α-řezů, jeho podstavu a jádro (případně vrchol). Počet α-řezů je členským parametrem instance výše přestavené servisní třídy, jakési továrny na fuzzy čísla, a je společný pro všechna fuzzy čísla, která jsou jejím prostřednictvím vytvořena. Výchozí hodnotou pro počet α-řezů je 11. Prostřednictvím továrny je do fuzzy čísel, respektive intervalů představujících jejich α-řezy, propagována také míra tolerance epsilon, řešící problém přesnosti binární reprezentace čísel s plovoucí řádovou čárkou. Výchozí hodnotou pro epsilon je 10^{-6}.

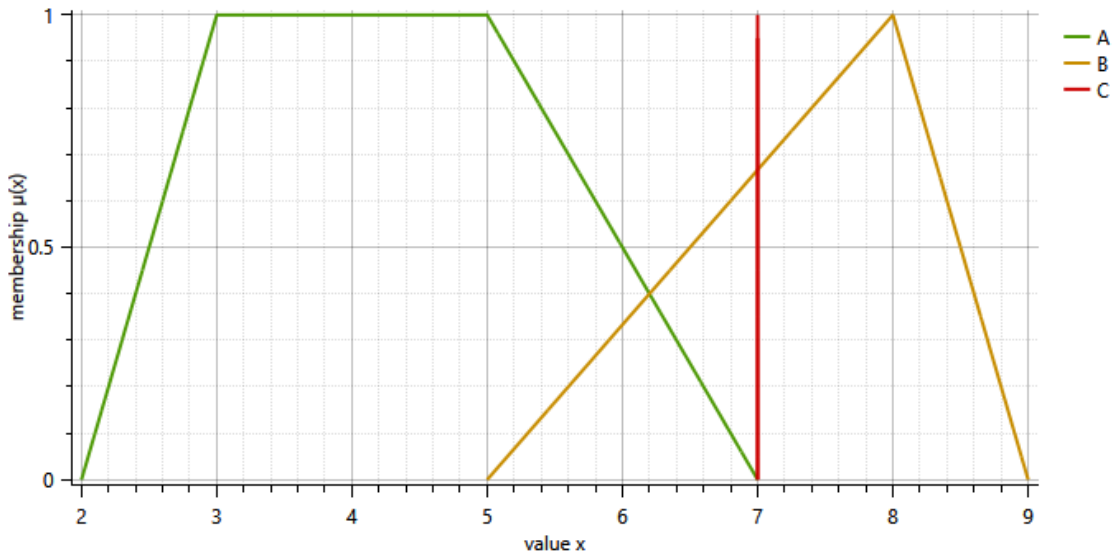
Třída `FuzzyNumberFactory` poskytuje celkem tři metody pro vytvoření nové instance `FuzzyNumber`, jakožto po částech lineárního fuzzy čísla. Metoda `CreateTrapezoidal()` slouží pro zadání fuzzy čísla pomocí jeho podstavy a jádra. Jako argumenty přijímá 4 čísla (`double`) jako hraniční body intervalů podstavy – $Supp$ a jádra – Ker, v pořadí $Supp_1$, Ker_1, Ker_2, $Supp_1$. Takto zadané fuzzy číslo má podobu lichoběžníku, proto je někdy označováno jako lichoběžníkové, trapezoidální, fuzzy číslo. Metody `CreateTriangular()` a `CreateCrisp()` pak nabízí o něco přímější způsob pro zadání speciálních případů lichoběžníkového fuzzy čísla, kdy jádro fuzzy čísla tvoří interval nulové délky, tedy $Ker_1 = Ker_2$, případně kdy je nulové délky i podstava fuzzy čísla.

Zdrojový kód 9 demonstruje vytvoření tří fuzzy čísel A, B a C pomocí továrny `FuzzyNumberFactory`. Všechna tři takto vytvořená fuzzy čísla budou mít shodně 6 α-řezů s rozlišovací schopností do 10^{-4}; přesně tak, jak je to specifikováno v konstruktoru dané továrny. Pro lepší představu jsou tato fuzzy čísla graficky znázorněna na Obrázek 4.

Zdrojový kód 9

```
var factory = new FuzzyNumberFactory(6, 1E-4);
FuzzyNumber A = factory.CreateTrapezoidal(2, 3, 5, 7);
FuzzyNumber B = factory.CreateTriangular(5, 8, 9);
FuzzyNumber C = factory.CreateCrisp(7);
```

Obrázek 4 – lichoběžníkové (A), trojúhelníkové (B) a ostré (C) fuzzy číslo



5.2 Implementace fuzzy aritmetiky

Matematicky je dokázáno, že libovolná operace nad fuzzy množinou, může být reprezentována jako tatáž operace nad množinou intervalů, jakožto množinou všech α-řezů dané fuzzy množiny [15]. Aritmetické operace s po částech lineárními fuzzy čísly lze tedy provádět jako soubor disktrétních operací nad kolekcí intervalů představujících α-řezy těchto fuzzy čísel.

Třída `FuzzyNumber` disponuje statickou metodou `Map()`, která umožňuje namapovat libovolnou unární operaci na kolekci intervalů konkrétní instance fuzzy čísla. Jako argument přijímá dané fuzzy číslo (instanci typu `FuzzyNumber`) a delegát funkce nebo přímo funkci (`Func<Interval, Interval>`), jež vrací instanci typu `Interval` a přijímá jediný předepsaný argument, kterým je `Interval`. Metoda `Map()` po zavolání vrátí novou instanci třídy `FuzzyNumber`, která je vytvořená právě z takto přemapované kolekce intervalů. Pro binární operace, tedy operace, ve kterých figurují dva operandy, tzn. dvě fuzzy čísla, existuje rovněž přetížená varianta metody `Map()`; viz Zdrojový kód 10.

Zdrojový kód 10

```
public static FuzzyNumber Map(FuzzyNumber X, Func<Interval, Interval>
operation)
{
    return new FuzzyNumber(X.AlphaCuts.Select(operation));
}
```

```
public static FuzzyNumber Map(FuzzyNumber X, FuzzyNumber Y,
Func<Interval, Interval, Interval> operation)
{
    if (X.AlphaCuts.Count == Y.AlphaCuts.Count)
    {
        return new FuzzyNumber(
            X.AlphaCuts.Zip(Y.AlphaCuts, operation)
        );
    }

    throw new NotImplementedException();
}
```

Současná implementace přetížené binární varianty metody `Map()` předpokládá shodný počet vstupních intervalů, respektive shodný počet α-řezů, které tvoří obě vstupní fuzzy čísla. V opačném případě metoda vyvolá výjimku. Důvodem je, že funkce, která zde reprezentuje nějakou binární operaci, musí být aplikovaná na korespondujících intervalech, tzn. α-řezech na shodném stupni příslušnosti. Pokud fuzzy čísla nemají shodný počet α-řezů, je zřejmé, že této podmínce nelze vyhovět. Řešením by samozřejmě bylo takovéto „chybějící" α-řezy vytvořit, například pomocí metody `GetAlphaCut()`, nicméně by to mělo neblahý dopad na výkon a efektivitu výsledné operace. Navíc díky továrně `FuzzyNumberFactory` lze předpokládat, že fuzzy čísla v rámci jedné aplikace budou tvořena téměř vždy se stejným počtem α-řezů. Proto lze současnou implementaci metody `Map()` v její přetížené variantě pro binární operace nad fuzzy čísly považovat za dostatečnou.

Základní aritmetické operace

Za základní aritmetické operace lze považovat binární operace pro sčítaní (+), odčítaní (-), násobení (×) a dělení (÷). Všechny tyto operace byly implementovány jako součást samotného fuzzy čísla, díky čemuž mohlo dojít k přetížení výše uvedených operátorů. Zdrojový kód 11 ukazuje implementaci přetíženého operátoru + pro sčítaní. Nejprve byl přetížen operátor u struktury `Interval`.

Zdrojový kód 11

```
public static Interval operator +(Interval left, Interval right)
{
    return new Interval(left.A + right.A, left.B + right.B,
        Math.Max(left.Epsilon, right.Epsilon)
    );
}

public static Interval operator +(Interval left, double right)
{
    return new Interval(left.A + right, left.B + right,
left.Epsilon);
```

```
}

public static Interval operator +(double left, Interval right)
{
    return right + left;
}
```

Pak bylo provedeno totéž pro třídu `FuzzyNumber`; viz Zdrojový kód 12. V ukázce je, mimo jiné, názorně vidět využití výše popisované metody pro mapování operací na kolekci intervalů fuzzy čísla – metody `Map()`. Podobně jsou implementovány i další základní aritmetické operace. Právě díky přetíženým operátorům může být práce s fuzzy čísly o něco bližší práci s klasickými ostrými čísly.

Zdrojový kód 12

```
public static FuzzyNumber operator +(FuzzyNumber left, FuzzyNumber
right)
{
    return Map(left, right, (x, y) => x + y);
}

public static FuzzyNumber operator +(FuzzyNumber left, double right)
{
    return Map(left, x => x + right);
}

public static FuzzyNumber operator +(double left, FuzzyNumber right)
{
    return right + left;
}
```

Ve výsledku lze s fuzzy čísly pracovat téměř jako s klasickými čísly standardních numerických typů (např. `double`). Jako příklad lze uvést jednoduchý aritmetický příklad, jehož zadání je patrné z kódu uvedeného níže (viz Zdrojový kód 13).

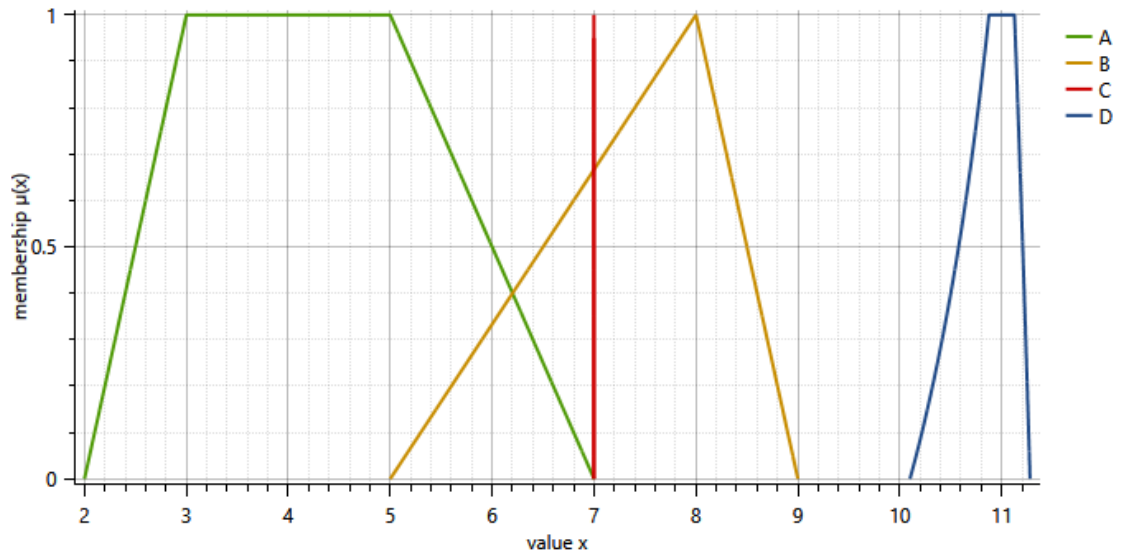
Zdrojový kód 13

```
var factory = new FuzzyNumberFactory();

var A = factory.CreateTrapezoidal(2, 3, 5, 7);
var B = factory.CreateTriangular(5, 8, 9);
var C = factory.CreateCrisp(7);

var D = 2 * C - (2.5 + A / B);
```

Zdrojový kód 13 navazuje na ukázku použití továrny na fuzzy čísla, servisní třídy `FuzzyNumberFactory`, viz Zdrojový kód 9. Poté, co jsou vytvořená čísla uložena do proměnných *A*, *B* a *C*, lze s nimi, díky přetíženým operátorům, pracovat zcela přirozeně a intuitivně. Přesně tak, jako s klasickými numerickými typy. Výsledek

aritmetické operace, hodnotu fuzzy čísla v proměnné *D*, graficky znázorňuje Obrázek 5.

Obrázek 5 – fuzzy aritmetika: D = 2 × C – (2,5 + A ÷ B)

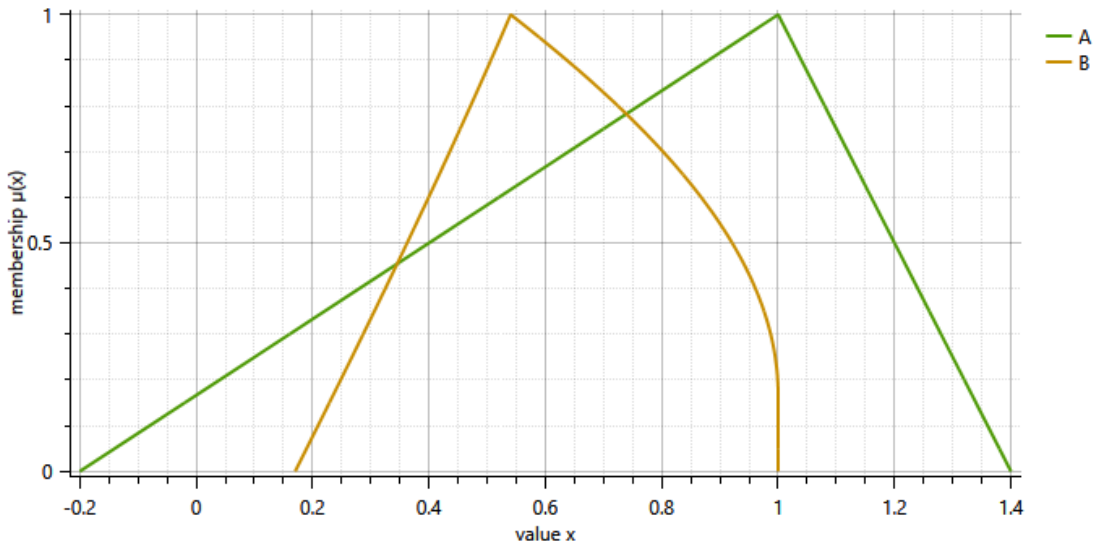


Vybrané funkce a operace nad fuzzy čísly

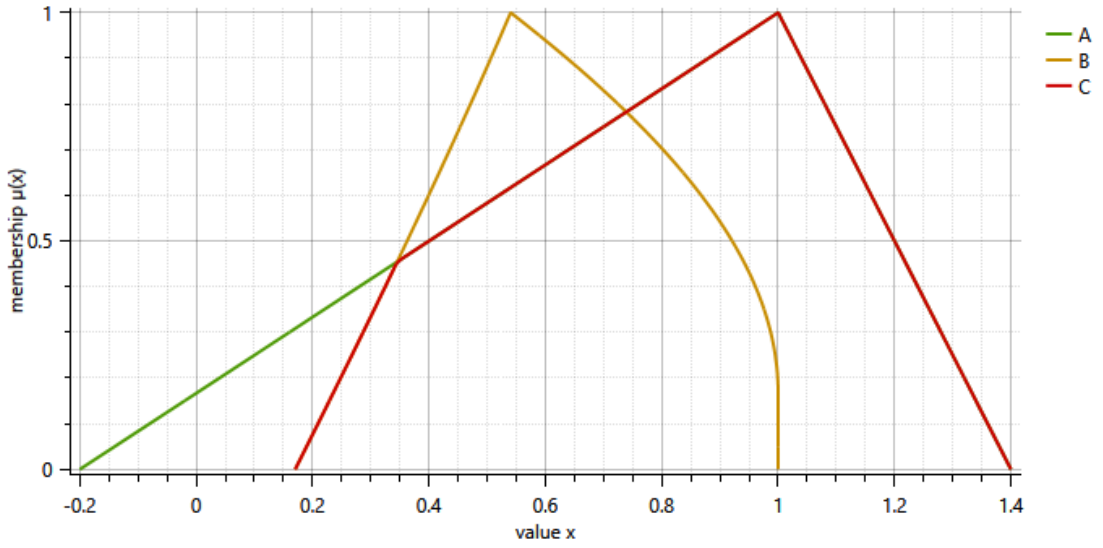
Je nutné říct, že při návrhu knihovny FuzzyMath šlo především o vytvoření pracovního rámce pro následnou implementaci aritmetických operací nad fuzzy čísly. Implementace konkrétních funkcí již nebyla prioritou, přesto několik takových vzniklo. Tyto aritmetické funkce jsou pak dostupné jako statické metody třídy `Functions`; jmenovitě to jsou

- goniometrické funkce sinus (sin), kosinus (cos), viz Obrázek 6, a tangens (tan);
- funkce atan2;
- funkce pro výpočet minima (min) a maxima (max), viz Obrázek 7, ze dvou různých fuzzy čísel;
- exponenciální funkce pow a exp.

Obrázek 6 – fuzzy čísla A, B; B = cos(A)



Obrázek 7 – fuzzy čísla A, B, C; C = max(A, B)



Porovnávání fuzzy čísel

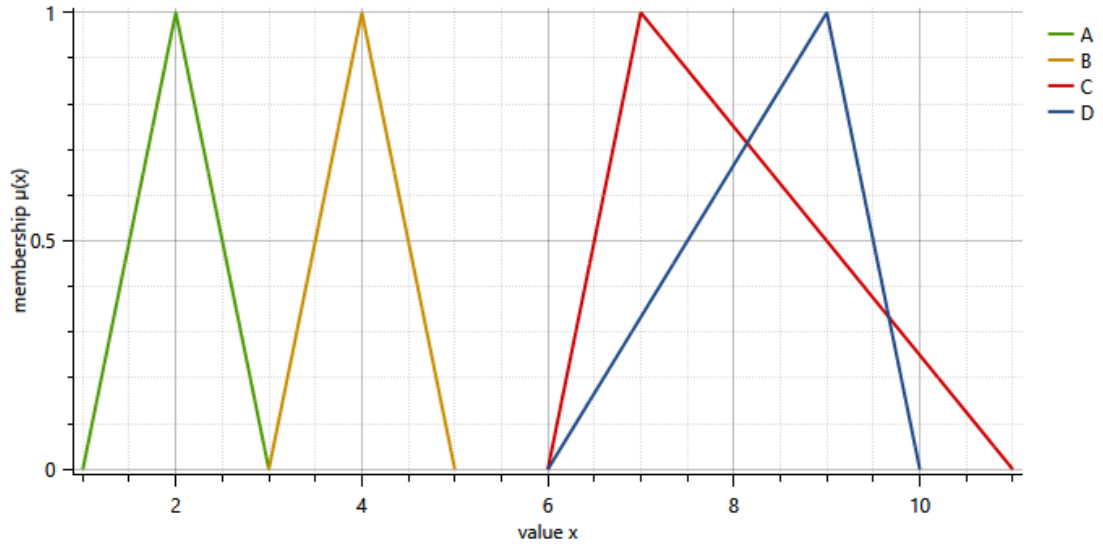
Porovnávání fuzzy čísel, ve smyslu větší/menší, je samo o sobě komplexním tématem. Existuje mnoho různých metod a způsobů, jak lze dvě fuzzy čísla vzájemně porovnávat, nicméně většina zvažuje jen určité parametry, zatímco jiné zanedbává [26].

Porovnávání dvou fuzzy čísel bylo implementováno v členské metodě třídy `FuzzyNumber`, metodě `GreaterThan()`. Ta jako argument přijímá další instanci fuzzy čísla a vrací hodnotu (`double`) z intervalu od 0 do 1, která vyjadřuje míru

jistoty v tvrzení, že instance fuzzy čísla, nad kterým byla metoda `GreaterThan()` volána, je větší, než ta předaná argumentem. Metoda je implementována tak, že je závislá na počtu α-řezů; čím více jich je, tím relevantnější by měla podávat výsledky.

Podobně jako v případě základních aritmetických operací (+ - × ÷), byly i v případě porovnávání dvou fuzzy čísel přetíženy operátory. Pro `a.Greather(b)` lze tedy použít ekvivalentní zápis `a > b`.

Obrázek 8 – fuzzy čísla A, B, C a D



Obrázek 8 ukazuje čtyři různá trojúhelníková fuzzy čísla A, B, C a D. Metoda `GreaterThan()` pak vrátí pro

- A > B hodnotu 0 – nulová jistota v tvrzení, že A je větší než B;
- B > A hodnotu 1 – maximální jistota v tvrzení, že B je větší než A;
- C > D hodnotu 0,545;
- D > C hodnotu 0,455;
- A > A hodnotu 0,5.

6 PRAKTICKÉ VÝSTUPY

Stěžejním výstupem praktické části této diplomové práce je softwarová knihovna pro reprezentaci fuzzy čísel a fuzzy aritmetiky, knihovna *FuzzyMath*. Hlavní motivací pro implementaci takové knihovny byl zoufalý nedostatek podobných nástrojů, schopných modelovat nejistotou a neurčitost v datech a pracovat s ní jako s přirozenou součástí těchto dat.

6.1 Knihovna FuzzyMath

FuzzyMath je softwarová knihovna vytvořená v programovacím jazyce C# nad platformou .NET ve verzi 4.0. Knihovna byla vytvořena v rámci diplomové práce „Implementace základních funkcí fuzzy aritmetiky v C# s využitím v ArcGIS“ na Katedře geoinformatiky na Univerzitě Palackého v Olomouci a následně byla uvolněna pod svobodnou licencí MIT.

Zdrojové kódy, včetně dokumentace, jsou publikovány ve veřejném Git repozitáři, dzibma/FuzzyMath [27], hostovaném na serveru GitHub. Obsah tohoto repozitáře je součástí digitální přílohy diplomové práce.

Vlastnosti knihovny FuzzyMath

Knihovna FuzzyMath umožňuje reprezentaci libovolného po částech lineárního fuzzy čísla jako uspořádanou kolekci jeho α-řezů. Hladiny stupňů příslušnosti jsou v rámci daného fuzzy čísla distribuovány rovnoměrně, a jsou tím pádem závislé na volbě počtu α-řezů.

Speciální typy po částech lineárních fuzzy čísel, jako jsou lichoběžníková a trojúhelníková fuzzy čísla, mohou být snadno vytvořeny z hraničních bodů podstavy a jádra (podstavy a vrcholu v případě trojúhelníkového fuzzy čísla).

Zvolený způsob reprezentace fuzzy čísel, jakožto po částech lineárních, umožnil přistoupit k fuzzy aritmetice jako k sérii odpovídacích aritmetických operací nad množinou, případně nad množinami, intervalů představujících jednotlivé α-řezy.

Knihovna samozřejmě podporuje základní aritmetické operace, kterými jsou sčítání (+), odčítání (-), násobení (×) a dělení (÷). Mimo to přináší implementaci některých vybraných funkcí jako je sinus, cosinus, tangens, funkce pro výpočet maximálního/minimálního fuzzy čísla z dvojice vstupních fuzzy čísel a některé další. Příklad práce s knihovnou FuzzyMath ukazuje Zdrojový kód 14.

Zdrojový kód 14

```
var factory = new FuzzyMath\FuzzyNumberFactory(6, 1E-5);

var a = factory.CreateTriangular(7, 7.2, 8);
var b = factory.CreateTriangular(6, 7, 9);

var c = 2 * a - a / b;

var max = FuzzyMath\Functions.Max(a, b);
var sin = FuzzyMath\Functions.Sin(a);
var pow = FuzzyMath\Functions.Pow(max, 2);

if ((a > c) > .5)
{
      // a is greater than c
}
```

Zvláštní důraz při návrhu a následném zpracovávání knihovny FuzzyMath byl kladen na univerzálnost a snadnou rozšiřitelnost. Výsledkem je minimalistické řešení okleštěné od zbytečných či okrajových funkčností, které by mělo sloužit především jako základ k vývoji dalších aplikací.

Instalace a použití

Knihovna FuzzyMath je dostupná prostřednictvím balíčkovacího systému NuGet, který slouží jako oficiální nástroj pro distribuci a správu softwarových balíčků a jejich závislostí v rámci vývojářské platformy .NET. Pro instalaci knihovny stačí do konzole pro správu balíčků, Package Manager Console, zadat a poté spustit následující příkaz.

```
PM> Install-Package FuzzyMath
```

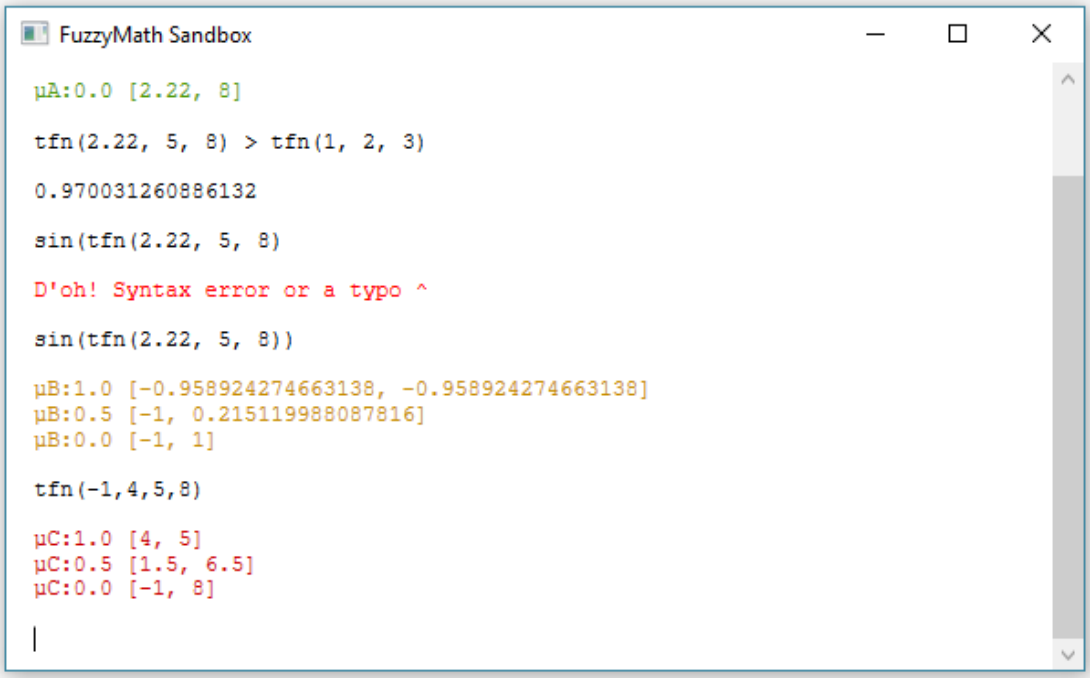
Případně lze využít dialogového okna pro správu závislostí, které je dostupné z kontextové nabídky v přehledu referencí projektu v rámci aplikace Visual Studio (Solution Explorer → References → Manage NuGet Packages…). Třídy knihovny FuzzyMath jsou pak připraveny k použití v rámci stejnojmenného jmenného prostoru (namespace) `FuzzyMath`.

6.2 FuzzyMath Sandbox

Při zpracovávání diplomové práce vzniknul nástroj pro vizualizaci fuzzy čísel a fuzzy aritmetiky – *FuzzyMath Sandbox*. Jedná se o malý jednosouborový program, který byl původně vytvořen pro testování a demonstraci možností knihovny FuzzyMath. To také vysvětluje jeho název, jednalo se o jakési pomyslné pískoviště

(angl. sandbox) pro hraní s fuzzy čísly a fuzzy aritmetikou. Teprve později, jako důsledek absence použitelných nástrojů na vizualizaci fuzzy čísel a fuzzy aritmetiky, bylo rozhodnuto, že i program FuzzyMath Sandbox bude zařazen mezi praktické výstupy této diplomové práce.

Obrázek 9 – uživatelské rozhraní FuzzyMath Sandbox



Uživatelské rozhraní

Uživatelské rozhraní programu je minimalistické (viz Obrázek 9) a tvoří jej jediné textové pole, které slouží jak pro zadávání vstupních příkazů, tak i pro případné textové odpovědi. Kromě několika příkazů není potřeba znát žádnou speciální syntaxi. Vstupní řetěz je odeslán ke zpracování stiskem klávesy Enter.

Vytvářet lze pouze lichoběžníková fuzzy čísla (a jejich speciální případy). Slouží k tomu metoda `tfn()` s argumenty oddělenými čárkou jako

a) posloupnost 4 čísel $Supp_1 \leq Ker_1 \leq Ker_2 \leq Supp_1$, jakožto hraničních bodů podstavy $Supp$ a jádra Ker daného fuzzy čísla;
b) posloupnost 3 čísel $Supp_1 \leq Ker_1 \leq Supp_1$, v případě, že $Ker_1 = Ker_2$;
c) jedno číslo, v případě, že $Supp_1 = Ker_1 = Ker_2 = Supp_1$.

Počet α-řezů, ze kterých budou fuzzy čísla zformována, lze zjistit pomocí metody `cuts()`. Tento počet lze případně změnit, a to pomocí stejné metody, `cuts()`, tentokrát však s počtem α-řezů jako argumentem.

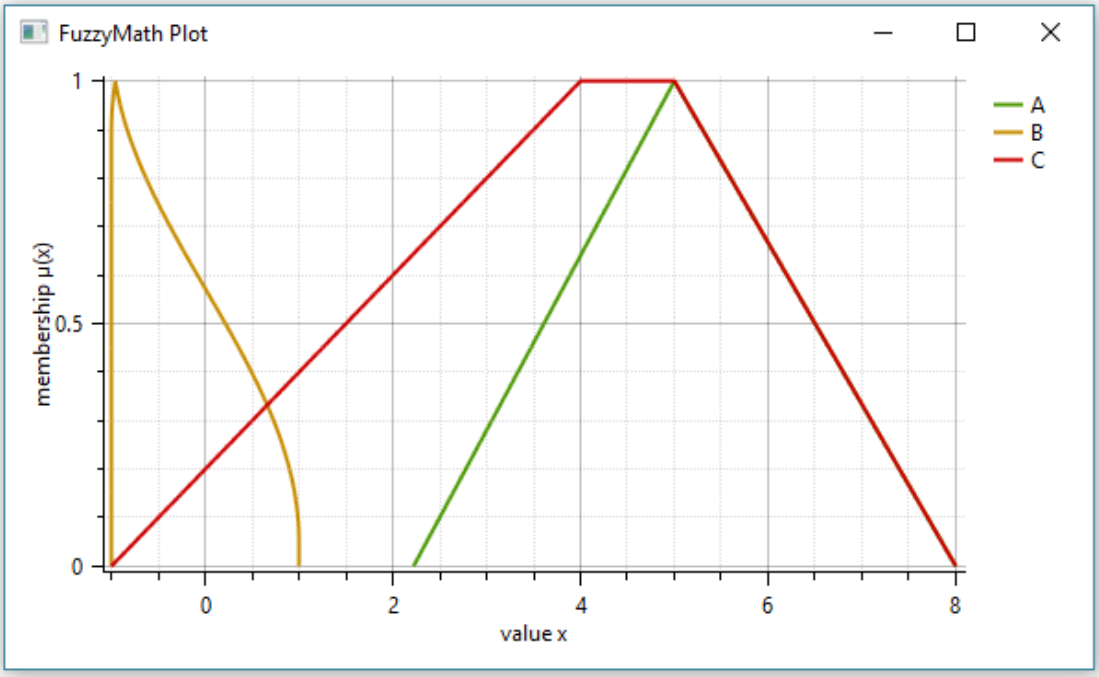
FuzzyMath Sandbox umožňuje počítat s (fuzzy) čísly pomocí známých aritmetických operátorů pro sčítání (+), odčítání (-), násobení (×) a dělení (÷). Pro práci s klasickými, tzn. ostrými, čísly lze využít matematické funkce poskytované v rámci jmenného prostoru .NET `System.Math`. Vyhodnocování číselných vstupů a práce s čísly se řídí stejnými pravidly jako práce s numerickými datovými typy v programovacím jazyce C#; tzn. je rozdíl mezi příkazem `5 / 3`, pro celočíselné dělení, a příkazem `5.0 / 3`.

Pro práci s fuzzy čísly lze využít aritmetické funkce implementované v rámci třídy `FuzzyMath.Functions`; konkrétně pak `sin()`, `cos()`, `tan()`, `atan2()`, `pow()`, `exp()`, `min()` a `max()`.

Grafický výstup

Pro vizualizaci fuzzy čísel slouží okno *FuzzyMath Plot*; viz Obrázek 10. Okno je inicializované automaticky, jakmile se na výstupu objeví první fuzzy číslo. Všechna fuzzy čísla jsou pak vykreslována do tohoto okna. Pokud je třeba okno s výstupy vyčistit, stačí ho jednoduše zavřít.

Obrázek 10 – vizualizace fuzzy čísel v okně FuzzyMath Plot



Exporty

FuzzyMath Sandbox podporuje export aktuálního obsahu okna FuzzyMath Plot do grafického formátu PNG. K vyvolání dialogového okna pro možnosti exportu slouží klávesová zkratka Ctrl+E nad aktivním oknem FuzzyMath Plot. Uživatel je vyzván k zadání názvu souboru s obrázkem exportovaného grafu. Může upravit jeho rozměry; ty ve výchozím stavu odpovídají velikosti grafu v okně FuzzyMath Plot. Kořenový adresář pro uložení vygenerovaného PNG souboru je adresář, ve kterém se nachází spustitelný soubor programu FuzzyMath Sandbox.

7 ILUSTAČNÍ PŘÍPADY VYUŽITÍ KNIHOVNY FUZZYMATH V GIS

Hlavním cílem této diplomové práce bylo navrhnout a vytvořit softwarovou knihovnu pro reprezentaci fuzzy čísel s možností základních aritmetických operací nad těmito čísly. Výsledkem práce, splňujícím takové zadaní, je knihovna *FuzzyMath* (viz kapitola 6.1). V následující kapitole pak budou předvedeny možnosti praktického využití této knihovny v prostředí GIS. Konkrétně budou popsány dva zásuvné (add-in) moduly pro ArcGIS for Desktop 10.2, využívající funkčnost knihovny FuzzyMath.

Záměrně byla zvolena případová studie na využití knihovny FuzzyMath ve spolupráci s komerční platformou ArcGIS (viz kapitola 2.2), jež je součástí stejnojmenného GIS ekosystému od firmy ESRI. Důvodem pro tuto volbu byl fakt, že ArcGIS v současné době představuje hlavní analytický a výukový nástroj na Katedře geoinformatiky na Univerzitě Palackého v Olomouci, kde tato práce vznikla.

Instalace ArcGIS for Desktop add-in

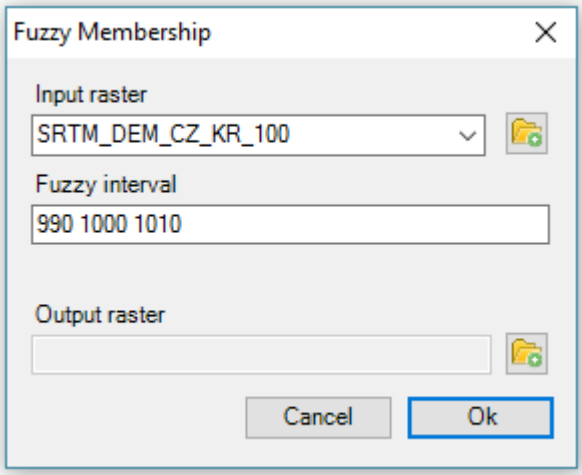
Instalace add-in modulu pro kteroukoliv z aplikací ArcGIS for Desktop je triviální záležitostí. V podstatě stačí dvojklikem „otevřít“ soubor modulu (soubor s koncovkou .esriAddIn), což vyvolá dialog pro instalaci daného add-in modulu. Ten pak již stačí jen potvrdit.

Po instalaci lze konkrétní add-in nalézt v aplikačním okně (ArcMap) *Customize* (Customize → Customize Mode…). V případě, že má doplněk vlastní toolbar, je po instalaci přidán do nabídky *Toolbars* (Customize → Toolbars). Výjimku představují doplňky vyžadující explicitní aktivaci v rámci extenzí dané aplikace.

7.1 FuzzyMembership ArcMap add-in

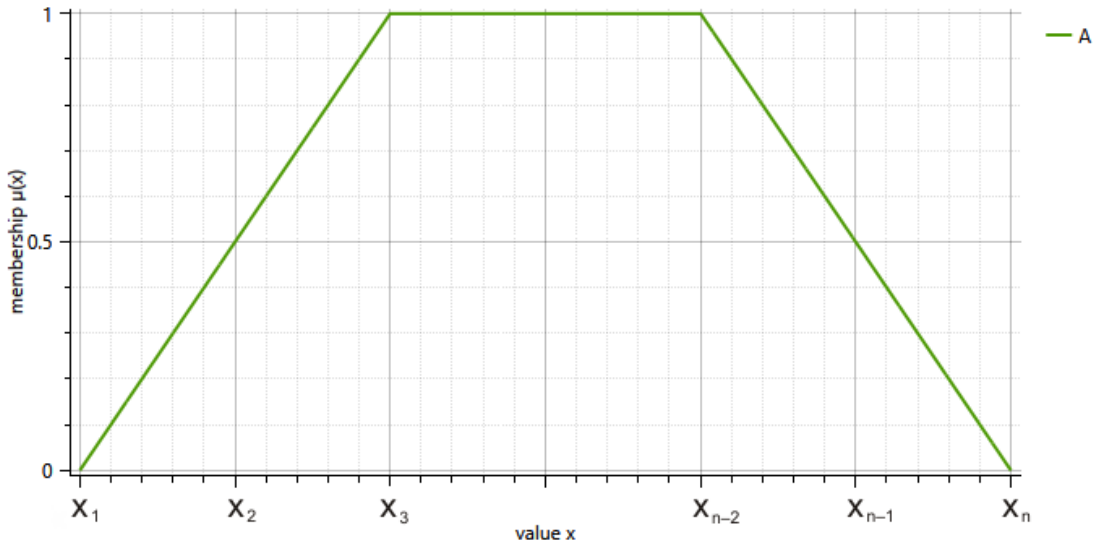
FuzzyMembership byl vytvořen jako ArcMap add-in pro fuzzifikovanou klasifikaci rastrů. Inspirací pro tvorbu tohoto add-in modulu byl stejnojmenný nástroj, který je součástí placené extenze ArcGIS for Desktop *SpatialAnalyst*. Přidanou hodnotou modulu FuzzyMembership je grafické uživatelské rozhraní a možnost snadného zprovoznění i na nejnižší licenční verzi produktu ArcGIS for Desktop. Po instalaci lze modul dohledat ve vlastním toolbaru FuzzyMembership.

Obrázek 11 – dialogové okno add-in modulu FuzzyMembership



Hlavní dialogové okno (viz Obrázek 11) zásuvného modulu FuzzyMembership souží pro zadání vstupních hodnot. Po uživateli vyžaduje tři vstupní parametry; vstupní rastr, hodnoty pro zadání fuzzy čísla a název klasifikovaného rastru. První a poslední parametr (vstupní a výstupní soubor) je zřejmý a nepotřebuje další komentář. Fuzzy číslo ve druhém parametru je zadáno z posloupnosti reálných čísel $x_1 \leq x_2 \leq \cdots \leq x_n$ oddělených mezerou. Z této posloupnosti se pak vytvoří po částech lineární fuzzy číslo tak, že jeho α-řezy budou intervaly $\langle x_1, x_n \rangle$, $\langle x_2, x_{n-1} \rangle$, $\langle x_3, x_{n-2} \rangle$, atd. (viz Obrázek 12).

Obrázek 12



Výstupem modulu FuzzyMembership je klasifikovaný rastr (např. viz Obrázek 13), který každému pixelu ze vstupního rastru přiřadí hodnotu od 0 do 1, která

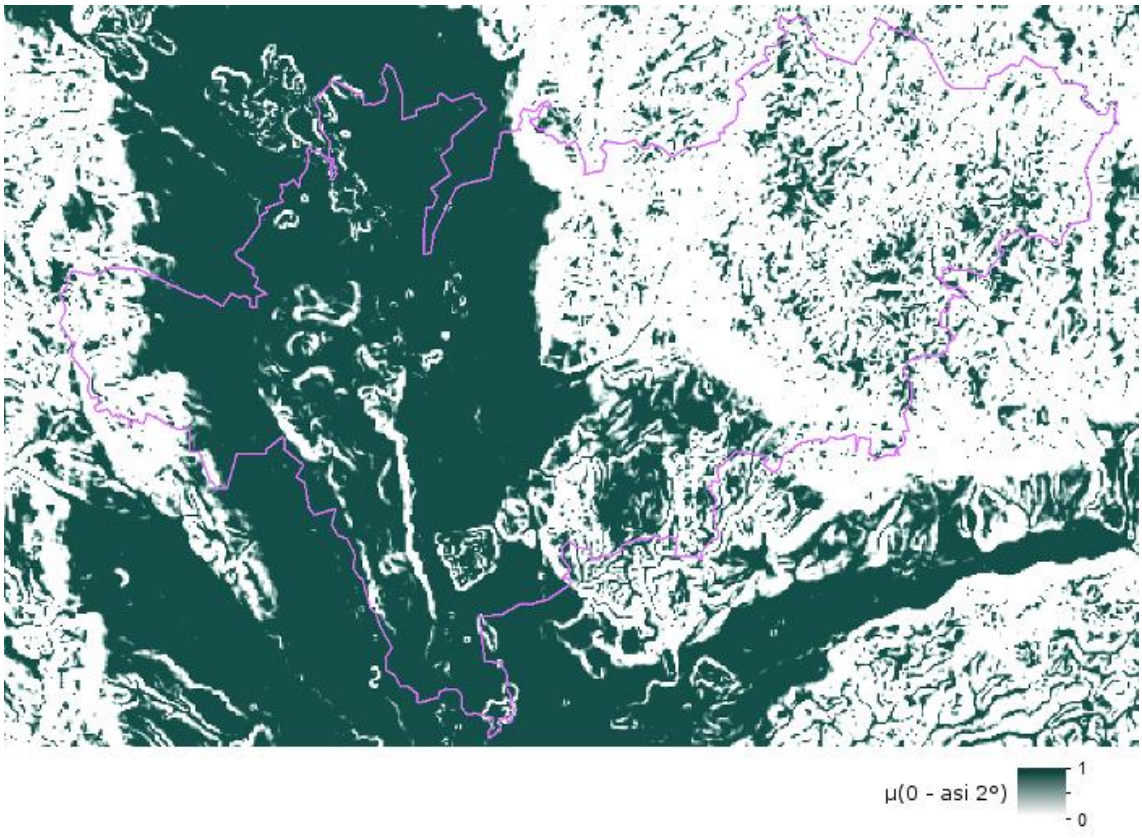
vyjadřuje stupeň příslušnosti k fuzzy číslu zadanému druhým parametrem v dialogovém okně uživatelského rozhraní FuzzyMembership.

Možnosti využití FuzzyMembership

Jako příklad pro využití modulu FuzzyMembership lze uvést fuzzy vyhledávání nebo dotazování v rastru. Lze díky němu poměrně snadno aplikovat vágní dotazy typu: „nadmořská výška kolem 1000 m“, „přibližně rovina“, „jižní svahy“ apod.

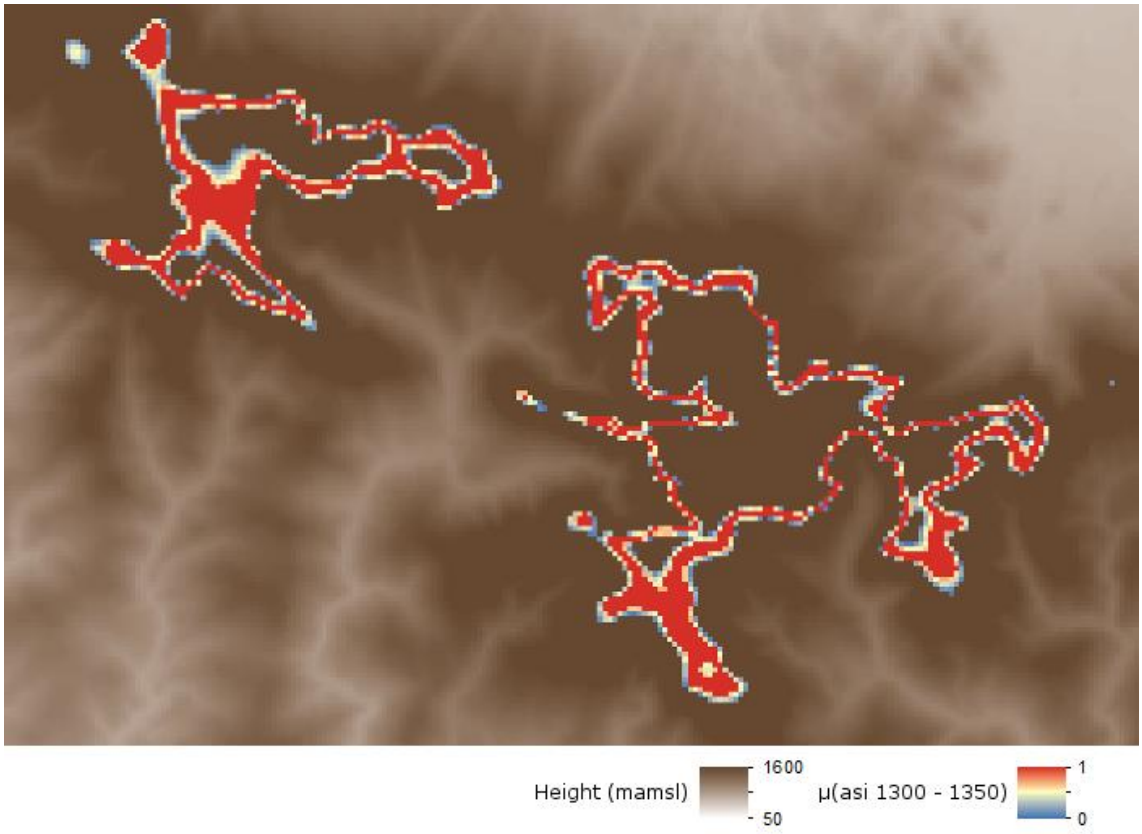
Pomocí fuzzy čísla na vstupu FuzzyMembership může být také modelována nejistota a nepřesnost v datech (např. vlivem maximální přesnosti konkrétní metody sběru dat), která bude následně zohledněna v rámci klasifikace rastru.

Obrázek 13 – terén do sklonu svahu asi 2° podle FuzzyMemberhip



Obrázek 13 zobrazuje možný výstup zásuvného modulu FuzzyMembership. V tomto konkrétním případě jde o výřez digitálního modelu sklonu terénu (rastr s hodnotami sklonu svahu ve stupních) nad ORP Olomouc, klasifikovaný podle hladiny příslušnosti do fuzzy intervalu 0° až asi 2°. Použitím fuzzy dotazu (tj. asi 2°) se do výsledku dostanou i ty varianty sklonu terénu, které by ostré klasifikaci nevyhověly, přesto že hranici 2° překračují jen minimálně; např. 2,01°.

Obrázek 14 – výsledek fuzzy dotazu podle FuzzyMembership



Obrázek 14 ukazuje další potenciální výstup modulu FuzzyMembership. Tentokrát jde o vágně zadaný dotaz, nad digitálním modelem terénu Krkonoš (SRTM). Dotaz by mohl být zadán slovně jako: „Najdi všechny body, které jsou v intervalu nadmořských výšek zhruba od 1300 do 1350 metrů." Mimo neurčitosti přítomné v samotném dotazu, může být volbou dotazovaného fuzzy intervalu zohledněna i nejistota způsobená přesností daného výškového modelu.

7.2 FuzzyRanking ArcMap add-in

FuzzyRanking je další add-in, který byl v rámci praktické části této diplomové práce vytvořen, aby demonstroval možnosti knihovny FuzzyMath v prostředí GIS. Opět se jedná o rozšíření aplikace ArcMap na bázi modelu pro psaní zásuvných modulů – *ArcObjects add-in*. Po instalaci lze modul FuzzyRanking dohledat ve stejnojmenném vlastním toolbaru.

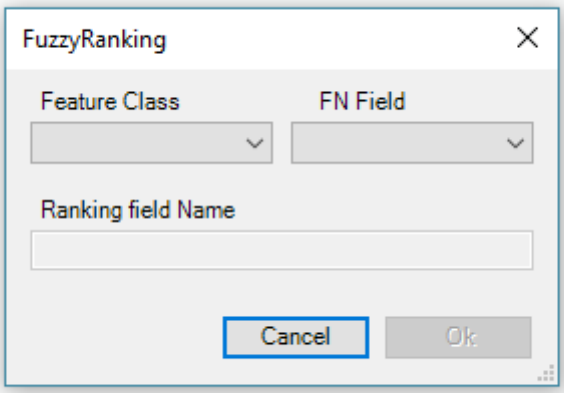
FuzzyRanking je modul, který se hodí na porovnávání vágně nebo nepřesně definovaných atributových hodnot s velmi malým rozptylem. Případně na nalezení potenciálních maxim mezi těmito hodnotami.

Add-in FuzzyRanking pracuje s vektorovou vrstvou, konkrétně pak s atributovou tabulkou jejích prvků. Smyslem FuzzyRanking je vyhodnotit horní pořadí prvků dané vektorové vrstvy, a to na základě konkrétního atributu v podobě po částech lineárního fuzzy čísla. Toto fuzzy číslo bude zadáno jako řetězec ve tvaru

- $Supp_1;Ker_1;Ker_2;Supp_2$ pro lichoběžníkové fuzzy číslo,
- $Supp_1;Ker_1;Supp_1$ pro trojúhelníkové fuzzy číslo ($Ker_1 = Ker_2$),
- a $Supp_1$ pro fuzzy číslo reprezentující ostrou hodnotu ($Supp_1 = Ker_1 = Ker_2 = Supp_1$),

kde $Supp_1$, Ker_1, Ker_2, $Supp_2$ představují reálná čísla (s desetinnou tečkou namísto čárky). Zároveň musí platit, že $Supp_1 \leq Ker_1 \leq Ker_2 \leq Supp_1$.

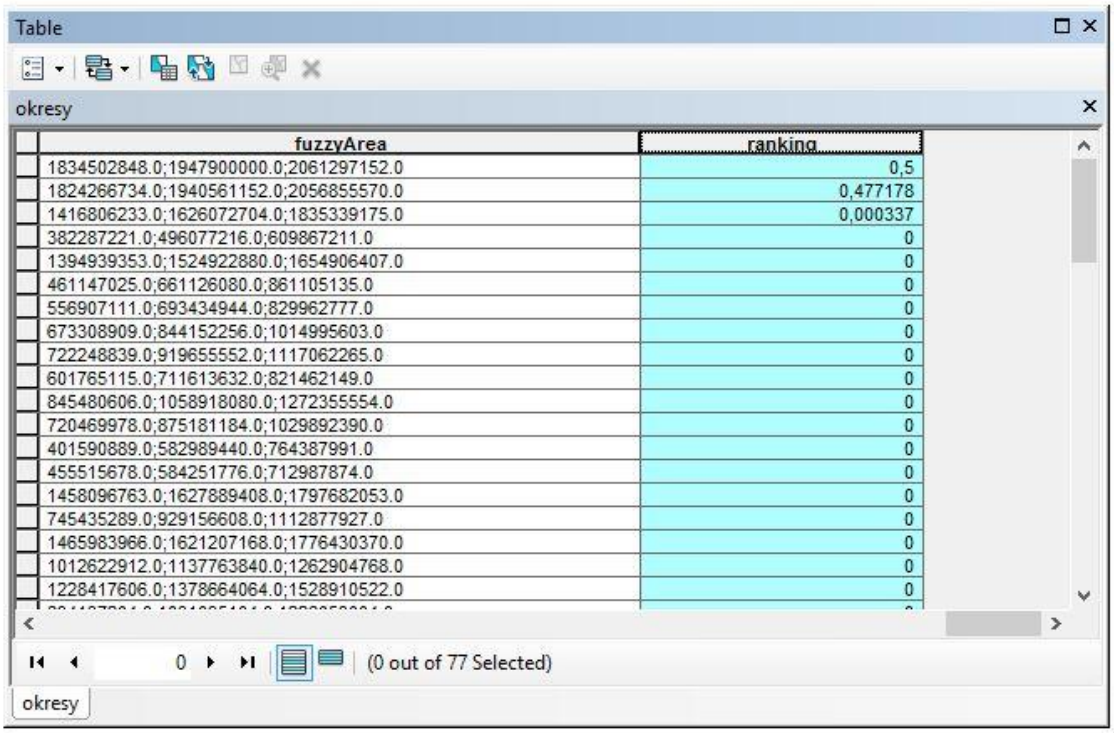
Obrázek 15 – dialogové okno add-in modulu FuzzyRanking



Obrázek 15 zobrazuje dialog pro zadání vstupních dat FuzzyRanking. Pokud v kontextové nabídce aktivní mapy existuje vektorová vrstva typu „Feature Class“, je nabídnuta ve stejnojmenném seznamu vrstev. Po výběru vstupní vrstvy jsou k výběru atributového pole s textovou reprezentací fuzzy čísla (viz výše) načteny všechny atributy typu „String“.

Pokud jsou data ve sloupci validní, FuzzyRanking je použije pro vytvoření fuzzy čísel $N_1, N_2, \ldots, N_k$, nad kterými vytvoří maximální hodnotu (maximální fuzzy číslo), jakožto $\max(N_1, N, \ldots N_k)$, kterou pak následně porovná s původními fuzzy čísly. Jako výsledek bude následně vytvořen nový sloupec, jehož hodnoty budou na intervalu od 0 do 1 vypovídat o míře jistoty v tvrzení $N_i > \max(N_1, N_2, \ldots N_k)$; např. viz Obrázek 16.

Obrázek 16 – atributová tabulka s výstupem FuzzyRanking



Získání vstupních „fuzzy“ dat

Při tvorbě add-in modulů nebylo cílem zpracovat nebo vyřešit konkrétní problém, ale pouze ukázat, jak je možné využít knihovnu FuzzyMath pro fuzzy aritmetiku v prostředí GIS. Pokud chybí nebo nejsou dostupná vhodná data, není problém si nějaká uměle vytvořit. K otestování funkčnosti add-in modulu FuzzyRanking lze využít například data vygenerovaná pomocí nástroje *FieldCalculator*, jenž je nativní součástí aplikace ArcMap. FieldCalculator umožňuje pro výpočet hodnoty pole využití uživatelsky definovaných funkcí; Zdrojový kód 15 (v jazyce Python) ukazuje příklad takovéto funkce.

Zdrojový kód 15

```
def generateFN(val, min, max):
    import random
    rnd = random.randint(min, max)
    return str(val - rnd) + ";" + str(val) + ";" + str(val + rnd)
```

Funkce `generateFN()` má tři parametry. Prvním je numerická hodnota, která bude uměle „rozostřena“ na fuzzy číslo. Další dva argumenty představují horní a dolní mez intervalu, ze kterého bude náhodně zvolena míra nepřesnosti. Návratovou hodnotou je žádaný řetězec (viz výše) popisující fuzzy číslo.

Možnosti využití FuzzyRanking

Prostor pro reálné využití modulu FuzzyRanking by mohl být všude tam, kde je třeba nalézt maximální hodnotu v sérii nepřesných měření. V případě vágně nebo nepřesně zadaných dat může být často obtížné najít jen jednu takovou (maximální) hodnotu. FuzzyRanking umožňuje nalézt a obodovat všechny potenciální maxima, a zároveň je odlišit od těch hodnot, která maximy být nemohou.

8 DISKUZE

Primárním cílem diplomové práce bylo vytvořit nástroj, který by umožnil reálnou aplikaci fuzzy aritmetiky a obecně fuzzy přístupu, při zpracování a analýze dat. Byla tedy naprogramována počítačová knihovna *FuzzyMath*. Knihovna je napsaná v jazyce C# na platformě .NET a je dostupná přes oficiální distribuční kanál této platformy, balíčkovací systém NuGet, pod svobodnou licencí MIT.

Zdrojové kódy knihovny byly vystaveny ve formě veřejného Git repozitáře na serverech populární webové služby GitHub. To spolu se zvolenou svobodnou licencí vytváří ideální prostor pro případnou údržbu a další rozvoj knihovny FuzzyMath, a to i nad rámec této diplomové práce.

Svým přístupem k implementaci fuzzy aritmetiky se knihovna FuzzyMath částečně podobá R balíčku FuzzyNumbers, vytvořeném pro statistické prostředí R. Ačkoliv oba řeší stejnou problematiku – fuzzy čísla a fuzzy aritmetiku, činí tak s docela jinou motivací. Zatímco balíček FuzzyNumbers představuje spíše koncové řešení pro výpočty a analýzy nad fuzzy čísly (pouze) v rámci statistického prostředí R, FuzzyMath je pak jakýsi pomyslný základní kámen pro vývoj libovolného softwaru využívajícího fuzzy čísla a fuzzy aritmetiku.

Dále byly, v rámci případové studie využití knihovny FuzzyMath v prostředí GIS, vytvořeny dva zásuvné (add-in) moduly pro desktopový GIS – ArcMap. Důvodem pro volbu komerční aplikace ArcMap, namísto open-source GIS, byl fakt, že představuje hlavní nástroj pro zpracování a analýzu prostorových dat používaný na katedře geoinformatiky v Olomouci, kde tato práce vznikala. Prvním takovým add-in modulem je *FuzzyMembership*, který umožňuje fuzzifikovanou klasifikaci rastru, dle příslušnosti k danému fuzzy číslu. Tím druhým je add-in modul *FuzzyRanking*, sloužící k hledání potencálních maxim mezi vágně nebo nepřesně zadanými hodnotami mezi atributy prvků vektorové vrstvy.

Je potřeba dodat, že oba naprogramované zásuvné moduly byly vytvořeny pouze za účelem demonstrovat možnost využití knihovny FuzzyMath v GIS, konkrétně v aplikaci ArcMap na platformě ArcGIS. Ani zdaleka nenaplňují obrovský potenciál, který fuzzy aritmetika a obecně fuzzy přístup k datům nabízí při práci s prostorovými daty a ve spojení s geoinformačními technologiemi. Ten lze hledat především v operacích a analýzách nad fuzzy povrchy [3, 4, 5, 6], síťových analýzách s fuzzifikovanými vstupy [7, 8] nebo fuzzy dotazech a při extrakci neurčitých informací z prostorových dat [9, 10].

Tato práce by měla být chápana jako první a nezbytný krok pro budoucí reálné aplikace fuzzy aritmetiky při práci (nejen) s prostorovými daty. Je důležité si

uvědomit, že každá operace či analýza, která má být fuzzifikována, vyžaduje pečlivou a časově náročnou přípravu. Každá taková analýza tak představuje potenciální zadání samostatné diplomové práce.

Knihovna FuzzyMath, jakožto primární výstup této práce, by pak měla být tím nástrojem, který poslouží jako základ při následné implementaci konkrétních fuzzifikovaných operací nebo analýz, a to nejen v prostředí GIS, ale všude tam, kde má práce s neurčitostí a nejistotou v datech smysl.

9 ZÁVĚR

V úvodu diplomové práce je zmíněn význam fuzzy přístupu k datům, respektive k neurčitosti a nejistotě, kterou data obsahují. Cílem této práce bylo vytvořit nástroj, který by umožnil neurčitost a nejistotu v datech modelovat a propagovat napříč operacemi a analýzami nad těmito daty. Vznikla proto otevřená knihovna *FuzzyMath* pro reprezentaci (po částech lineárních) fuzzy čísel a implementující fuzzy aritmetiku nad těmito čísly. Knihovna je napsána v programovacím jazyce C# nad platformou .NET a uvolněná pod svobodnou licencí MIT. Zdrojové kódy jsou dostupné ve veřejném Git repozitáři, dzibma/FuzzyMath, populární webové služby GitHub. Jeho kopie je součástí digitální přílohy práce (viz Seznam příloh).

Samotnému vývoji knihovny předcházela relativně náročná výzkumná část, kdy bylo nutné nastudovat a pečlivě promyslet možné přístupy k návrhu fuzzy čísel, fuzzy aritmetiky a k jejich následné implementaci. Všechna podstatná rozhodnutí jsou popsána, případně zdůvodněna, v textu práce.

V rámci případové studie možnosti využití knihovny FuzzyMath v prostředí GIS byly vytvořeny dva zásuvné (add-in) moduly pro ArcGIS for Desktop, konkrétně pro aplikaci ArcMap – *FuzzyMembership* a *FuzzyRanking*. Oba moduly jsou, včetně zdrojových kódů, součástí digitální přílohy práce (viz Seznam příloh).

Zároveň s knihovnou FuzzyMath vzniknul i nástroj pro vizualizaci po částech lineárních fuzzy čísel a aritmetiky nad těmito čísly – aplikace *FuzzyMath Sandbox*. Jedná se o jednoduchý, nicméně pro daný účel velice efektivní nástroj, pomocí kterého, mimo jiné, byly vytvořeny grafické vizualizace fuzzy čísel a fuzzy aritmetiky obsažené v textu této práce. FuzzyMath Sanbox je jednosouborová aplikace, kterou není třeba nijak instalovat; samotná aplikace, včetně zdrojových kódů je součástí digitální přílohy práce (viz Seznam příloh).

Vytvořením knihovny FuzzyMath a dvou zásuvných modulů pro aplikaci ArcMap (FuzzyMembership a FuzzyRanking), jako ukázky možností využití této knihovny v prostředí GIS, byly splněny všechny cíle vytyčené na začátku této diplomové práce.

LITERATURA A INFORMAČNÍ ZDROJE

[1] M. F. McNeill a . E. Thro , Fuzzy Logic: A Practical Approach, AP Professional, 1994.

[2] W. Fellin, H. Lessmann, M. Oberguggenberger a R. Vieider, Analyzing Uncertainty in Civil Engineering, Berlin: Springer, 2005, pp. 51-72.

[3] J. Caha, „Uncertainty propagation in fuzzy surface analyses“, UP Olomouc, Olomouc, 2014.

[4] J. Caha, L. Marek a J. Dvorský, „Predicting PM10 Concentrations Using Fuzzy Kriging“, v *Hybrid Artificial Intelligent Systems*, Bilbao, 2015.

[5] P. Fisher a J. Caha, „On Use of Fuzzy Surfaces toDetect Possible Elevation Change“, v *GIScience*, Vienna, 2014.

[6] S. Soltani-Mohammadi, „FuzzyKrig: a comprehensive matlab toolbox for geostatistical estimation of imprecise information“, *Earth Science Informatics,* pp. 1-11, 15 Říjen 2015.

[7] J. Caha a J. Dvorský, „Optimal Path Problem with Possibilistic Weights“, *Geoinformatics for Intelligent Transportation,* Lecture Notes in Geoinformation and Cartography, pp. 39-50, 2014.

[8] M. Ghatee a M. S. Hashemi, „Application of fuzzy minimum cost flow problems to network design under uncertainty“, *Fuzzy Sets and Systems,* sv. 160, č. 22, pp. 3263-3289, 2009.

[9] J. Caha a J. Dvorský, „Querying on Fuzzy Surfaces with Vague Queries“, v *Hybrid Artificial Intelligent Systems*, Spain, 2013.

[10] H. Yu, S.-w. Hwang a . K. C.-C. Chang, „Enabling soft queries for data retrieval“, *Information Systems,* sv. 32, č. 4, pp. 560-574, 2007.

[11] „Microsoft Developer Network“, Microsoft, [Online]. Dostupné z https://msdn.microsoft.com/cs-cz/default.aspx. [Přístup získán 20. března 2016].

[12] T. Nash, Accelerated C# 2010, Apress, 2010.

[13] „Esri - GIS Mapping Software, Solutions, Services, Map Apps, and Data“, ESRI, [Online]. Dostupné z http://www.esri.com/. [Přístup získán 15. března 2016].

[14] „ArcObjects Help for .NET developers“, ESRI, [Online]. Dostupné z http://desktop.arcgis.com/en/arcobjects/latest/net/webframe.htm. [Přístup získán 15. března 2016].

[15] M. Hanns, Applied Fuzzy Arithmetic: An Introduction with Engineering Applications, Springer, 2005.

[16] „MATLAB: Fuzzy Logic Toolbox“, MathWorks, [Online]. Dostupné z http://www.mathworks.com/products/fuzzy-logic/. [Přístup získán 1. dubna 2016].

[17] Wolfram, „Fuzzy Logic Documentation“, [Online]. Dostupné z http://media.wolfram.com/documents/FuzzyLogicDocumentation.pdf. [Přístup získán 1. dubna 2016].

[18] M. Gagolewski a J. Caha, „. FuzzyNumbers Package: Tools to deal with fuzzy numbers in R,“ [Online]. Dostupné z https://github.com/Rexamine/FuzzyNumbers/raw/master/devel/tutorial/FuzzyNumbers-Tutorial_current.pdf. [Přístup získán 1. dubna 2016].

[19] L. A. Zadeh, „Fuzzy Sets,“ Department of Electrical Engineering and Electronics Research Laboratory, University of California, 1965.

[20] D. Dubois a H. Prade, Fuzzy Sets and Systems: Theory and Applications, Academic Press, 1980.

[21] W. A. Klimke, „Uncertainty Modeling using Fuzzy Arithmetic and Sparse Grids,“ Universität Stuttgart, Stuttgart, 2006.

[22] A. Kaufmann a G. M. Madan , Introduction to Fuzzy Arithmetic: Theory and Applications, Van Nostrand Reinhold Company, 1985.

[23] R. Baekeland a E. E. Kerre, „Piecewise linear fuzzy quantities: A way to implement fuzzy information into expert systems and fuzzy databases“, *Uncertainty and Intelligent Systems,* sv. 313, Lecture Notes in Computer Science, pp. 119-126, 1988.

[24] R. E. Moore, Interval Analysis, Prentice-Hall, 1966.

[25] R. E. Moore, B. R. Kearfott a M. J. Cloud, Introduction to interval analysis, Society for Industrial and Applied Mathematics, 2009.

[26] D. Dubois a H. Prade, „Ranking fuzzy numbers in the setting of possibility theory“, *Information Sciences,* sv. 30, č. 3, pp. 183-224, 1983.

[27] „dzibma/FuzzyMath,“ GitHub, [Online]. Dostupné z https://github.com/dzibma/FuzzyMath. [Přístup získán 3. května 2016].

SEZNAM PŘÍLOH

Příloha č. 1: Kompaktní disk

Obsahuje

- projekt knihovny FuzzyMath;
- aplikaci FuzzyMath Sandbox, včetně zdrojových kódů v C#;
- zásuvné moduly pro aplikaci ArcMap – FuzzyMembership a FuzzyRanking; včetně zdrojových kódů v C#;
- webové stránky s prezentací diplomové práce;
- text diplomové práce v elektronické podobě.