VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

VYTVOŘENÍ INTERAKTIVNÍCH PROGRAMŮ PRO PODPORU VÝUKY ZPRACOVÁNÍ SIGNÁLŮ A OBRAZŮ

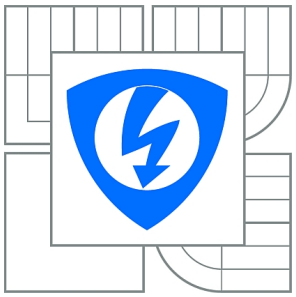
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAROSLAV OLBERT
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

VYTVOŘENÍ INTERAKTIVNÍCH PROGRAMŮ PRO PODPORU VÝUKY ZPRACOVÁNÍ SIGNÁLŮ A OBRAZŮ

INTERACTIVE SOFTWARE TOOLS FOR TEACHING SIGNAL AND IMAGE PROCESSING

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

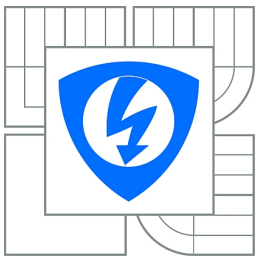
AUTOR PRÁCE — JAROSLAV OLBERT

AUTHOR

VEDOUCÍ PRÁCE — Ing. MARIE DAŇKOVÁ

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student: Jaroslav Olbert ***ID:*** 154822
Ročník: 3 ***Akademický rok:*** 2014/2015

NÁZEV TÉMATU:

Vytvoření interaktivních programů pro podporu výuky zpracování signálů a obrazů

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvořit programy pro interaktivní podporu výuky zejm. v kurzech BZSG, MGMP, BASS. Budou mít podobu JAVA či Flash apletů. Aplety budou tématicky zaměřeny na: 1/ lineární kombinaci obrazů, 2/ metodu nejmenších čtverců a lineární regresi, 3/ doporučení mířícího bodu při házení šipkami na terč pomocí dvojrozměrné konvoluce. V rámci BP navrhněte jejich funkcionalitu a uživatelské rozhraní a následně je implementujte ve zvoleném programovacím jazyce.

DOPORUČENÁ LITERATURA:

[1] Žára, J.; Beneš, B.; Sochor, J.; Felkel, P.: Moderní počítačová grafika (2. vydání), Computer Press, 2005, ISBN 80-251-0454-0.
[2] Anděl, J.: Matematická statistika. 1. vyd. Praha: SNTL/ALFA, 1978.
[3] Schildt, H.: Java7, výukový kurz, Computer Press, Brno, 2012.

Termín zadání: 9.2.2015 ***Termín odevzdání:*** 2.6.2015

Vedoucí práce: Ing. Marie Daňková
Konzultanti bakalářské práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Bakalářská práce se zabývá tvorbou Java apletů pro podporu výuky zpracování signálů a obrazů. Teoretická část práce se krátce věnuje historii a výhodám jazyka Java, následně pak teoriemi k jednotlivým apletům a popisem zdrojových kódů, kterými jsou aplety tvořeny. Praktickou část tvoří návrh a tvorba apletů pro lineární kombinace obrazů, metodu nejmenších čtverců a doporučení mířicího bodu při hodu šipkami na terč.

KLÍČOVÁ SLOVA

Java, aplet, obraz, lineární kombinace, metoda nejmenších čtverců, Gaussovo rozdělení, házení šipek, míření

ABSTRACT

The bachelor's thesis deals with creating Java applets to support teaching signal and image processing. The theoretical part of the thesis briefly describes the history and advantages of the Java language; subsequently the theories for individual applets and descriptions of the source codes are presented. The practical part consists of the design and creation of applets for linear combinations of images, the least squares method and recommendation of an aiming point on the darts target.

KEYWORDS

Java, applet, image, linear combination, least squares method, Gaussian distribution, darts throwing, aiming

OLBERT, Jaroslav *Vytvoření interaktivních programů pro podporu výuky zpracování signálů a obrazů*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2015. 52 s. Vedoucí práce byla Ing. Marie Daňková

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Vytvoření interaktivních programů pro podporu výuky zpracování signálů a obrazů“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucí bakalářské práce slečně Ing. Marii Daňkové za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále bych rád poděkoval svým rodičům za podporu při studiu.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Tato práce se zabývá návrhem a tvorbou interaktivních programů pro podporu výuky předmětů jako jsou např. Analýza signálů a soustav, Základy počítačové sazby a grafiky, Moderní počítačová grafika. Tyto programy slouží jako doplnění teoretického výkladu, který bývá často nezáživný a pro studenty nezajímavý. Programy slouží především k vytvoření představ, jak daná problematika funguje v praxi. Výuka laboratorních úloh funguje rovněž velmi dobře, avšak ne vždy je zcela dostačující a je téměř nemožné provádět tuto výuku mimo školní prostředí. Jelikož dnes je již přítomnost výpočetní techniky a připojení k internetu v každé domácnosti téměř samozřejmostí, mohou si tak studenti pomocí těchto aplikací sami vyzkoušet, jak daná problematika funguje v praxi, a to i z pohodlí domova.

Obsahem této práce je teoretický rozbor a následná implementace několika zadaných témat. Práce je rozdělena do částí, kdy každá část se zabývá jedním z témat. V každé části je popsána teorie dané problematiky a dále pak obsahuje popis samotného apletu. Veškeré výsledné programy a soubory jsou přiloženy v příloze.

První aplet slouží k prezentaci lineární a konvexní kombinace obrazů. Obsahuje tři obrazy, u kterých lze volit hodnotu, kterou se násobí veličiny jednotlivých pixelů daného obrazu. Uživatel má možnost volit mezi lineární a konvexní kombinací a rovněž má možnost výběru z několika obrazů ve stupních šedi. Tyto obrazy pak může libovolně kombinovat a násobit vybranými hodnotami. Výsledný obraz se poté zobrazí ve spodní části apletu.

Druhý aplet, slouží k výpočtům a názornému předvedení aproximace bodů v rovině pomocí metody nejmenších čtverců. Aplet obsahuje plátno, které slouží k vykreslování zadaných bodů, ale především k vykreslení výsledné funkce. Dále obsahuje tabulku pro body, které uživatel zadá a ovládací prvky, pomocí nichž se aplet ovládá.

Poslední, třetí, aplet se zabývá výpočtem a zobrazením doporučeného mířicího bodu při házení šipkami na terč. Cílem je, při zadaných parametrech, uživateli doporučit místo na terči, na které, pokud bude mířit, získá hodem průměrně největší možný počet bodů. Tento aplet využívá pro výpočet mířicího místa Gaussova rozdělení ve dvojrozměrném prostoru a dále dvojrozměrnou konvoluci.

1 VÝBĚR PROGRAMOVACÍHO JAZYKA

Danou problematiku lze řešit několika způsoby. Aplety je možné tvořit pomocí programovacího jazyka ActionScript, který se používá k vytváření webových aplikací a animací, které poté tvoří Flash aplikace, nebo v tomto případě aplety. Dalším způsobem řešení je použití programovacího jazyku Java. Tento jazyk jsem zvolil i já a pokusím se zde odůvodnit, proč jsem učinil právě toto rozhodnutí.

1.1 Programovací jazyk Java

Programovací jazyk Java vznikl v roce 1995 [9, 2] a ihned po svém uvedení si získal množství příznivců. Dá se říci, že společnost Sun Microsystems, která s tímto programovacím jazykem přišla, způsobila zásadní krok kupředu v oblasti programování. Díky Javě došlo ke znatelné změně Webu, který tak dostal interaktivní prostředí, které je pro uživatele na ovládání příjemnější. Během následujících let doznala Java několika vylepšení a pokračovala tak v růstu dále. Během těchto let se Java postupně vyvinula z původní verze 1.0 na dnešní verzi 1.8. V dubnu ruku 2009 začal proces, kdy firmu Sun Microsystems získala společnost Oracle. Tento proces skončil až téměř po roce, v lednu 2010. Vývoj jazyka Java tedy od tohoto data má plně v rukou společnost Oracle, která stojí za vydáním verze 1.7 také označované jako Java SE 7, která přinesla mnohá vylepšení.

V dnešní době patří Java k nejrozšířenějším a nejpoužívanějším programovacím jazykům na světě a je téměř nemyslitelné, aby zařízení připojené k internetu neobsahovalo některou z jejích verzí. Díky tomu se Java stále vyvíjí a přizpůsobuje širšímu okruhu uživatelů a zařízení. Příkladem může být využití Javy v mobilních zařízeních se systémem Android, kdy Java slouží jako hlavní nástroj ve tvorbě aplikací pro tyto přístroje [2].

Právě díky vlastnostem, jako je například všestrannost, popularita, podpora a jednoduchost Javy, je tento jazyk stále více populárnější a rozšířenější. V budoucnu se bude jistě dále vyvíjet a také díky tomu se bude neustále vylepšovat prostřednictvím novým verzím.

Osobně jsem se rozhodl použít, při vytváření apletů pro tuto práci, právě tento programovací jazyk z důvodů výše uvedených. Dále také proto, že již mám alespoň základní znalosti tohoto programovacího jazyka a práce v tomto prostředí mi přijde poměrně jednoduchá, na to, jak složité programy se s jeho pomocí dají vytvořit. Jazyk ActionScript jsem si nevybral z toho důvodu, že s ním nemám doposud žádné zkušenosti a díky informacím od některých spolužáků se mi zdá, že oproti Javě je i značně složitější.

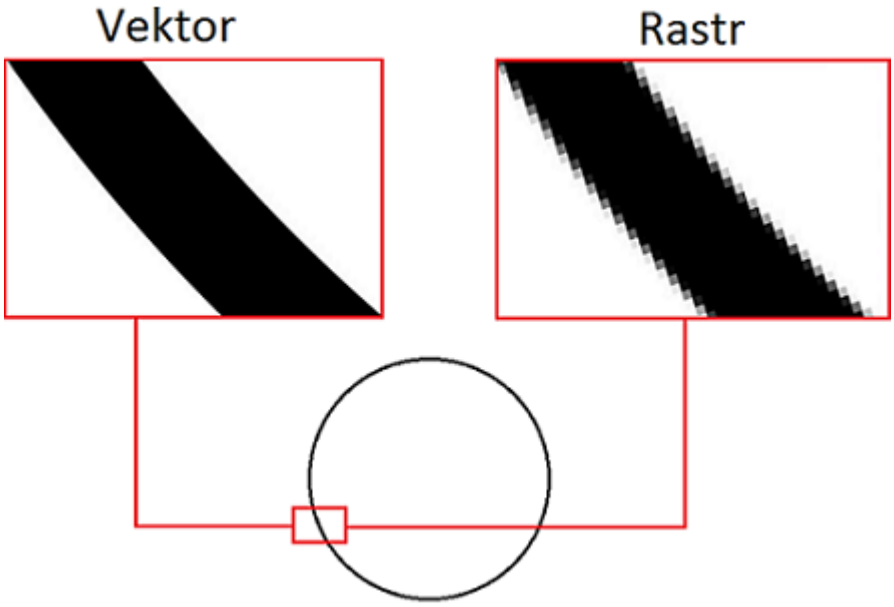
2 LINEÁRNÍ KOMBINACE OBRAZŮ

2.1 Teoretická část

Obrazy v počítačové grafice Pokud se budeme zabývat počítačovou grafikou, musíme ji rozdělit na dvě části:

- Vektorová grafika
- Rastrová grafika

Vektorovou grafiku tvoří objekty, které lze matematicky popsat pomocí rovnic. Takovými objekty jsou např. úsečka, křivka, bod a jiné. Tyto objekty poté vytváří výsledný obraz, který uživatel vidí. Výhodou vektorové grafiky je, že výsledné soubory jsou v podstatě pouze textové soubory, které obsahují rovnice vložených objektů a díky tomu jsou velmi malé [7].



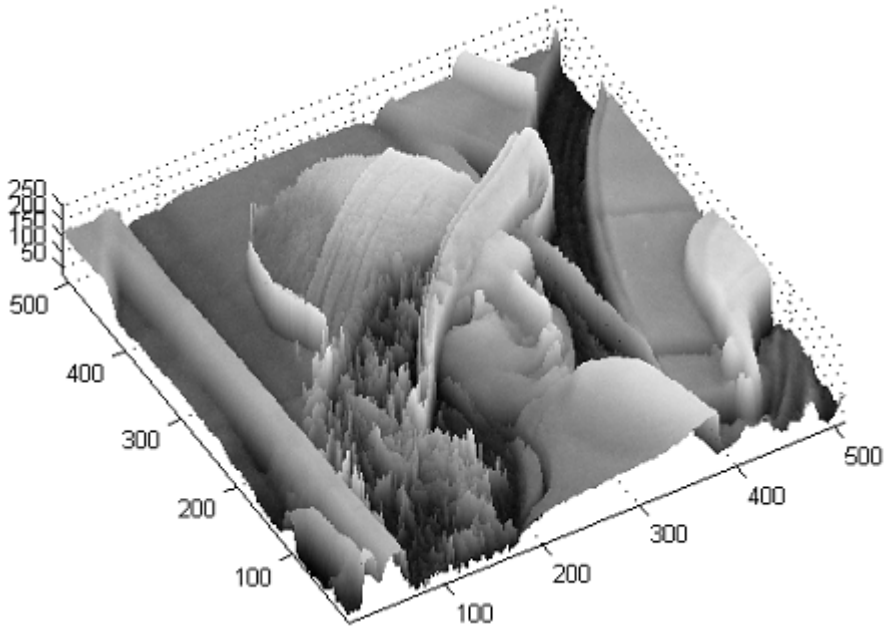
Obr. 2.1: Rozdíl mezi rastrovou a vektorovou grafikou.

Rastrová grafika, na rozdíl od vektorové, je tvořena mřížkou, nebo chceme-li maticí, kde každý jednotlivý prvek má svou barvu. Díky tomu, že lidské oko nedokáže rozpoznat rozdíly mezi tak malými body, splývají hrany bodů do sebe a tvoří iluzi dokonalého obrazu. Samozřejmě čím více se budeme pokoušet obraz zvětšit, tím více budou hrany bodů viditelnější a tím bude obraz méně kvalitní. Body, o kterých se bavíme, jsou nazvány *pixely*, což je zkratka výrazu „picture element“, který můžeme do češtiny přeložit jako prvek obrazu. Jak již bylo řečeno, pixel v rastrové grafice tvoří bod v obraze a je charakteristický svojí barvou.

Matematicky bychom mohli tento jev vyjádřit pomocí matice A s $m \times n$ prvky, která by byla popsána rovnicí $A[m, n] = h$, kde m a n označují řádek a sloupec matice, a h značí barevnou hodnotu daného prvku matice.

Na obrázku 2.3 můžeme názorně vidět, jak by vypadalo zobrazení barevných hodnot obrazu ve 3D. Na osách X a Y vidíme hodnoty souřadnic jednotlivých pixelů v obraze. Na ose Z pak najdeme hodnotu odstínu pixelu ve stupních šedi, resp. hodnotu jasu. Tyto hodnoty nabývají velikosti od 0 (černá barva) do 255 (bílá barva), případně od 0 do 1 (záleží na intervalu, se kterým pracujeme).

Obr. 2.2: Původní obrázek.



Obr. 2.3: 3D znázornění hodnot barevné škály původního obrázku. Jedná se o stejný obraz jako 2.2 [8].

Barvy v počítačové grafice tvoří obrazy, ať už se jedná o vektorovou, nebo rastrovou grafiku. Veškeré barvy jsou tvořeny kombinací několika základních barev podle vybraného barevného modelu. Model RGB tvoří tři základní složky - červená, zelená a modrá, které nabývají hodnot $\langle 0, 1\rangle$ nebo $\langle 0, 255\rangle$, podle toho s jakým intervalem pracujeme. Pomocí kombinací těchto složek lze získat jakýkoliv barevný odstín [10, 7].

2.1.1 Lineární kombinace v algebře

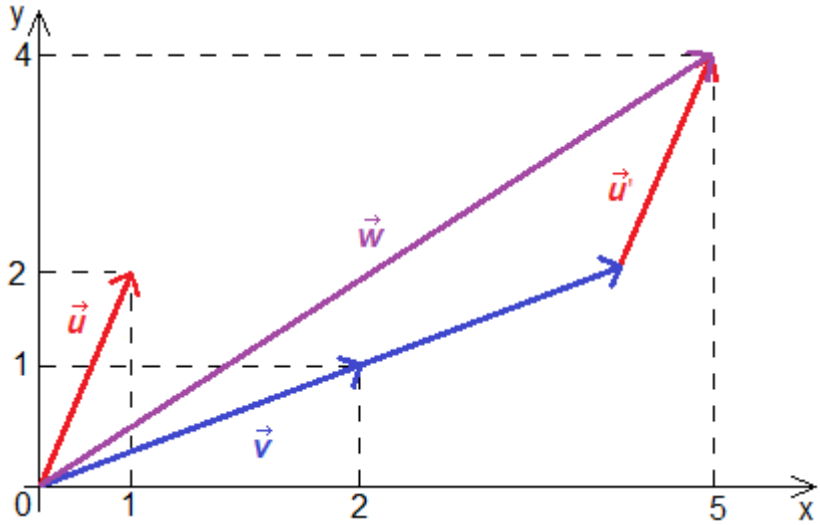
Matematická definice lineární kombinace je v podstatě zobecnění pojmů sčítání a násobení pro čísla. Můžeme ji zapsat pomocí rovnice $x = a_i \cdot x_i + \cdots + a_k \cdot x_k = \sum_{i=1}^{k} a_i \cdot x_i$, kde x_i je zadaný prvek, který chceme lineárně upravovat, a_i je číslo z množiny reálných čísel, kterým násobíme a k označuje počet prvků.

Pro představu mějme tři vektory $\vec{x}$, $\vec{y}$, $\vec{z}$. Pokud vektor $\vec{x}$ nelze vyjádřit lineární kombinací vektorů $\vec{y}$ a $\vec{z}$, stává se lineárně nezávislým, to znamená, že žádnou matematickou operací nemůžeme vektory $\vec{y}$ a $\vec{z}$ upravit tak, aby jejich součet byl roven $\vec{x}$ [1].

Na obrázku 2.4 můžeme vidět příklad lineární kombinace vektorů. Červenou barvou je znázorněn vektor $\vec{u}$ a modrou barvou vektor $\vec{v}$. Fialovou barvou je pak znázorněn vektor $\vec{w}$, který vznikl součtem těchto vektorů $\vec{w} = \vec{u} + 2 \cdot \vec{v}$. Výpočet poté vypadá následovně:

$$\vec{u} = (1, 2), \vec{v} = (2, 1), \tag{2.1}$$

$$\vec{w} = (1, 2) + 2 \cdot (2, 1) = (5, 4). \tag{2.2}$$



Obr. 2.4: Lineární kombinace vektorů.

2.1.2 Konvexní kombinace

Konvexní kombinace je speciální případ lineární kombinace, pokud platí, že všechny koeficienty lineární kombinace jsou nezáporné a zároveň je jejich součet roven jedné. Platí tedy vztah $\sum_{i=1}^{k} a_i = 1; 1 \geq a_i \geq 0$, kde k označuje množinu vektorů z daného vektorového prostoru a a_i označuje k-tici čísel z tělesa [3].

2.1.3 Lineární kombinace obrazů

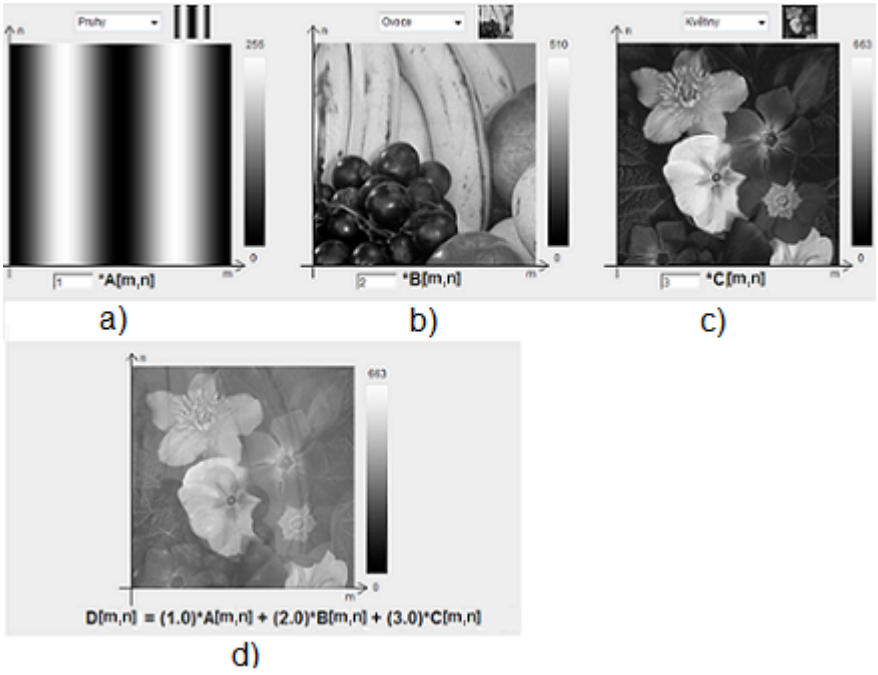
Lineární kombinaci obrazů můžeme chápat z matematického hlediska jako součet vynásobených prvků více obrazů. Toto tvrzení by se dalo vyjádřit pomocí vztahu $\sum_{i=1}^{k} a_i \cdot x_i$, kde a_i značí hodnotu, kterou obraz násobíme, a x_i označuje příslušný obraz.

Pokud se bavíme o lineární kombinaci obrazů v počítačové grafice, [10] ji popisuje jako nejjednodušší postup přeměny několika obrázků v jeden výsledný obraz. Dochází zde k tzv. přetavení obrazu a tento postup je založen na interpolaci barvy pixelu. Jednoduše dochází k tomu, že obrázky se postupně prolínají za pomocí úpravy jednotlivých RGB složek daných obrázků. U konvexní kombinace rozlišujeme kromě RGB složek také Alpha složku, která určuje průhlednost obrazu. Nabývá hodnot od 0 do 100% u každého pixelu zvlášť. Tento koeficient také označujeme jak koeficient průhlednosti pixelu. Následně tedy máme namísto trojice RGB čtveřici RGBα, nebo lze také použít zápis RGBA. Pomocí koeficientu α můžeme měnit průhlednost jednotlivých pixelů, ale i celého obrazu. Zároveň jej ale můžeme chápat dvěma způsoby:

- Pixel obsahuje barevné složky RGB a je průhledný z α procent.
- Složky RGB pokrývají pixel pouze z α procent.

Tohoto se využívá např. pro antialiasing.

Na obrázku 2.5 si můžeme prohlédnou lineární kombinaci obrazů v praxi na výstupu z vytvořeného apletu. Vidíme zde 3 obrazy a), b), c), které slouží jako vstupy, a výsledný obraz d), který znázorňuje výstupní obraz.



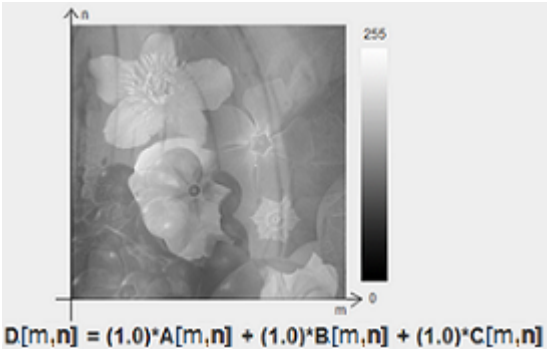
Obr. 2.5: Příklad lineární kombinace obrazů použité v apletu. a) Vstupní obraz A, b) vstupní obraz B, c) vstupní obraz C, d) výstupní obraz, tedy lineární kombinace obrazů A, B, C popsaná rovnicí $D = 1 \cdot A + 2 \cdot B + 3 \cdot C$.

2.2 Popis rozhraní apletu

Aplet je vytvořen v programovacím jazyce Java. Demonstruje, jak probíhá lineární kombinace obrazů v počítačové grafice a rovněž zobrazuje i výsledek kombinace. Tento aplet obsahuje čtveřici obrázků. Tři z nich, nacházející se v horní části apletu, slouží ke zobrazení vybraného obrazu z palety možností a zároveň slouží i pro vykreslení již vynásobených obrazů. Pod každým z těchto obrazů se nachází textové pole, do kterého uživatel zadává hodnoty pro násobení. Vedle každého z trojice vykreslených modifikovaných obrazů se nachází škála znázorňující stupně šedi s hodnotami minimální a maximální hodnoty barevných složek obrazu. Ve spodní části je umístěn obraz výsledného součtu modifikovaných obrazů a dále nabídka pro výběr mezi lineární a konvexní kombinací. Rozložení apletu je zobrazeno na obrázku 2.12.

2.2.1 Ovládání apletu

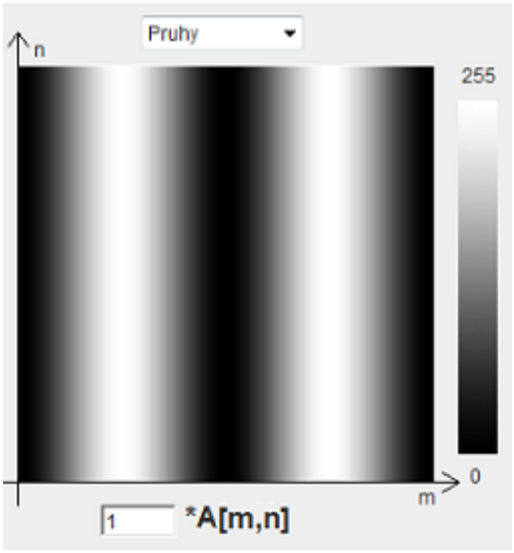
Ovládání apletu je velmi jednoduché a přehledné. V horní části si uživatel, pomocí kurzoru myši, zvolí libovolný z nabízených obrázků. Ve střední části pak do textového pole, pod daným obrazem, zadá hodnotu, kterou chce násobit jednotlivé pixely daného obrázku a stiskne *Enter*. Je důležité po každé změně hodnot v textovém poli



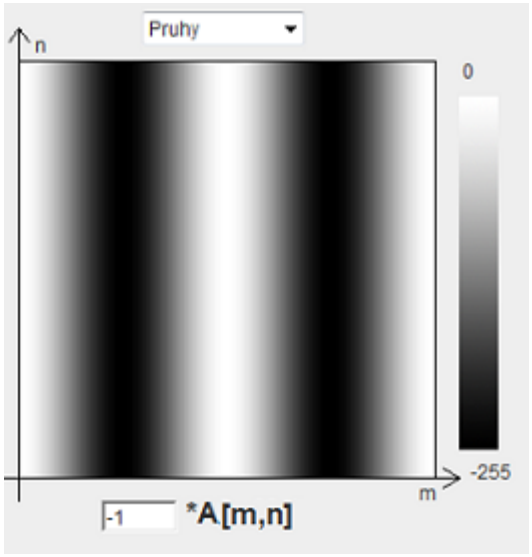
Obr. 2.6: Příklad výstupu apletu při násobení stejnými hodnotami.

Obr. 2.7: Příklad výstupu apletu při násobení jednou hodnotou výrazně vyšší.

a při volbě jiného obrazu z palety možností potvrdit změnu opětovným stisknutím klávesy *Enter*, jinak se změny neprojeví.



Obr. 2.8: Násobení obrazu kladnou hodnotou.



Obr. 2.9: Násobení obrazu zápornou hodnotou.

Pokud je v textovém poli zadán neplatný znak (něco jiného než číslo), vypíše se chyba. Jako desetinné znaménko je potřeba používat tečku. Čárka bude brána jako chybný znak. Ve spodní části apletu se, po zadání všech tří hodnot do textových polí, zobrazí výsledný součet upravených obrazů.

Ve spodní části apletu má uživatel na výběr zvolení lineární, nebo konvexní kombinace obrazů. Jako výchozí je nastavena lineární kombinace. Po přepnutí na konvexní kombinaci se pod textovými poli zobrazí nápověda s rozsahem hodnot, ze kterého by uživatel měl vybrat příslušnou hodnotu. Pokud uživatel dosud žádnou z hodnot nezadal, je automaticky nastaven rozsah možných hodnot od $0,0$ do $1,0$.

Po vyplnění jednoho z polí se nápověda u ostatních dvou přizpůsobí a doporučený interval se zmenší o hodnotu již zadanou. Obdobný postup následuje při zadání druhé ze tří hodnot, dokud není splněna podmínka, že součet všech tří hodnot je roven jedné. Pokud tato podmínka splněna není, zobrazí se chybová hláška, která uživatele upozorní na chybu.

Obrázky 2.6 a 2.7 znázorňují příklady výstupů apletu při různých hodnotách násobení obrazů. Na obrázku 2.6 je vidět, že všechny tři vstupní obrazy jsou přítomny ve stejné míře a výsledný obrázek tak obsahuje všechny tři původní obrazy. Obrázek 2.7 znázorňuje výstup apletu, když je některá ze zadaných hodnot výrazně vyšší, než ty ostatní. Na výstupu je tedy poté vidět pouze obrázek s touto vysokou hodnotou.

2.3 Popis funkcí ve zdrojovém kódu

Samotný aplet je napsán ve třídě *Projekt1*, která je vložena do balíku projekt1, jenž je součástí projektu *Linearni Kombinace*. Samotná třída je rozšířena o metodu *JApplet*, díky níž samotný aplet může vzniknout.

V hlavní metodě *Projekt1()* se nachází veškeré funkce apletu. Celá metoda se skládá z několika cyklů, které jsou prováděny v závislosti na akci uživatele. V úvodu jsou vloženy funkce, které naslouchají změně při volbě obrázku z nabídky v horní části apletu. Názvy jednotlivých obrázků jsou pomocí konstrukce *switch()* převedeny na typ string, se kterým se poté dále pracuje.

V následující sekci kódu již dochází ke zpracování samotných obrazů. Jsou vytvořeny tři shodné funkce pro každý jednotlivý zvolený obrázek. Tyto funkce opět probíhají v cyklech a jsou volány pokaždé, když program zaznamená reakci uživatele. V úvodu funkce je získaná hodnota, zadaná do textového pole, převedena na typ double a přiřazena do proměnné. Pokud dojde k zadání neplatného znaku, tedy jiného znaku než čísla a desetinné tečky, je automaticky uživateli zobrazena chybová hláška.

Po úspěšném získání zadané hodnoty dochází k násobení jednotlivých barevných složek RGB pixelů obrazu zadanou hodnotou. Zároveň je zaznamenána nejvyšší maximální hodnota, která označuje nejsvětlejší odstín obrazu. Jelikož pracujeme s obrázky pouze ve stupních šedi, jsou hodnoty všech tří složek RGB stejné a tudíž je jedno, kterou z barevných složek pro určení maxima použijeme. V tomto případě byla použita složka červené barvy.

K získání jednotlivých barevných složek z obrazu slouží následující řádek

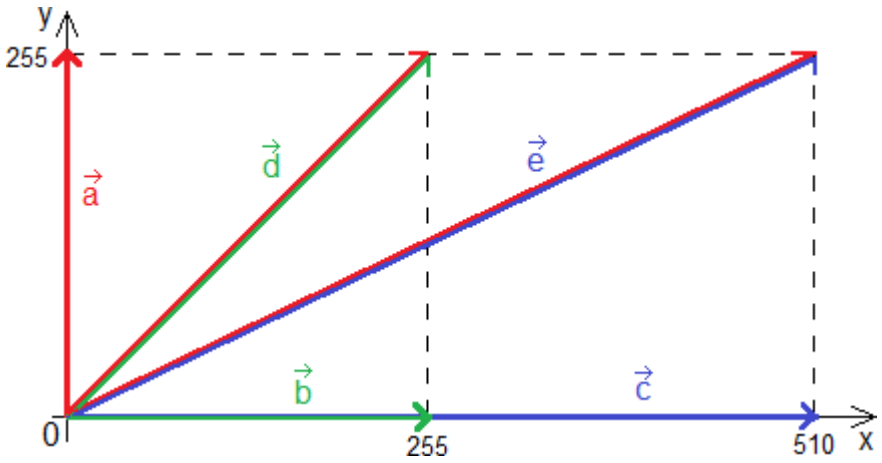
```
Color c = new Color(image1.getRGB(j, i));
```

kde j a i označují pozici pixelu v obraze.

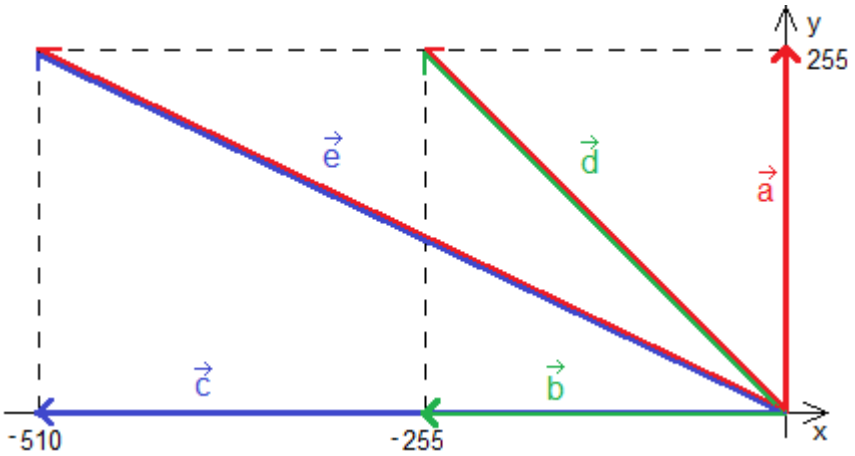
Po získání maximální hodnoty dochází k lineárnímu škálování obrazu, aby nedošlo k přetečení hodnot, k čemuž se využívá vztah

$$y = kx + q. \tag{2.3}$$

V tomto vztahu označuje y hodnoty, které potřebujeme získat, x hodnoty, které jsme získali násobením, k koeficient, kterým je potřeba hodnoty x vynásobit, abychom získali příslušné hodnoty y. Škálování v našem případě slouží pouze pro vykreslování obrazů, pokud by došlo k přetečení hodnot. Příklad můžeme vidět na obrázku 2.10 a 2.11.



Obr. 2.10: Příklad použití lineárního škálování, použitého v apletu.

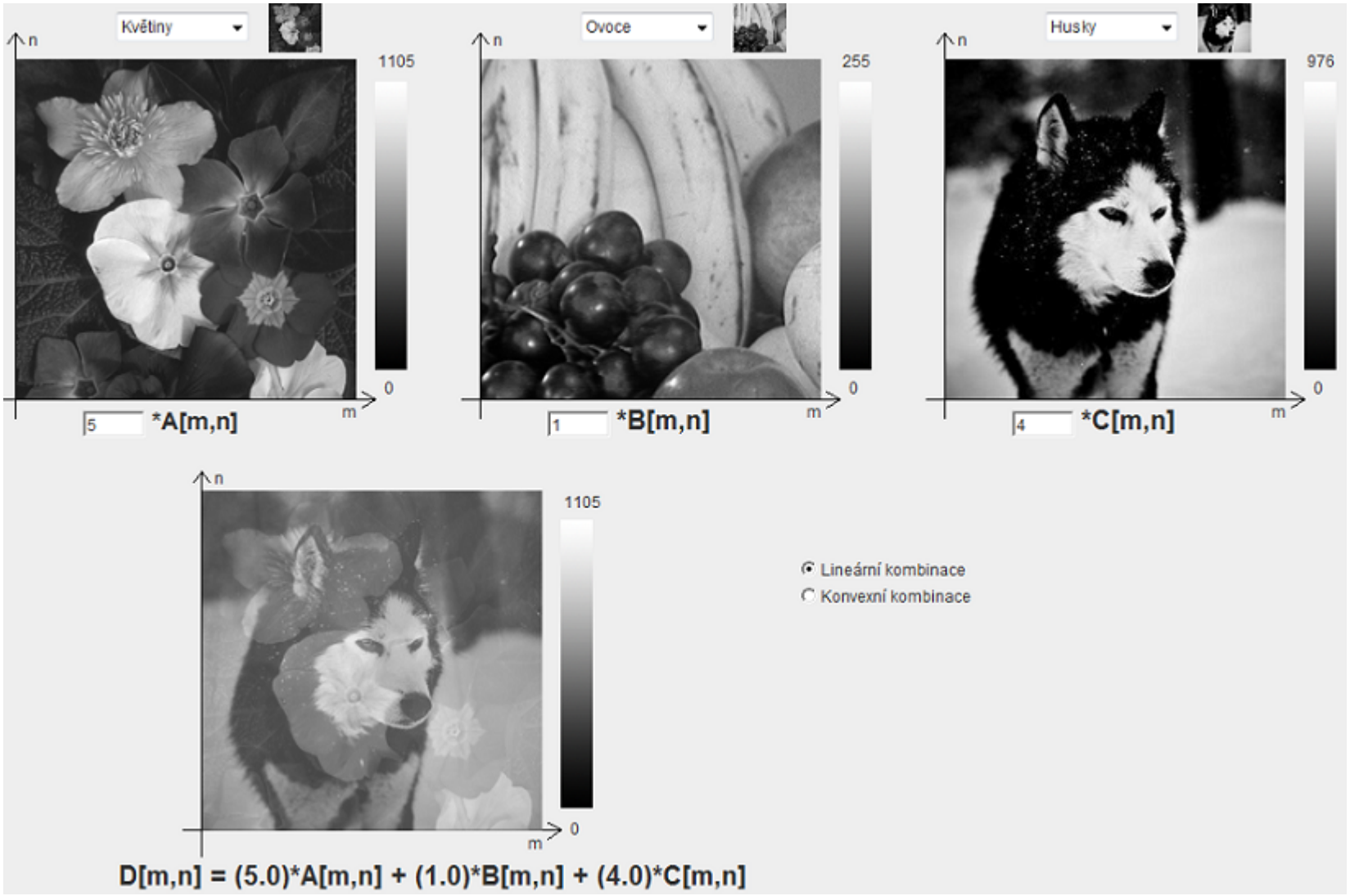


Obr. 2.11: Lineární škálování pro záporné hodnoty.

Na obrázcích vidíme červeně vektor $\vec{a}$, který potřebujeme získat, zeleně vektor $\vec{b}$, který dostaneme, když vynásobíme obraz 1x a maximální hodnota je tedy 255, modře vektor $\vec{c}$, který je dvojnásobkem vektoru $\vec{b}$. Vektory $\vec{d}$, $\vec{e}$ naznačují, jak probíhá změna hodnot, které máme, na hodnoty, které potřebujeme získat. Obrázek 2.11 znázorňuje stejnou funkci, pouze pro záporné hodnoty.

Následně dochází k překreslení původního obrazu na nový, již lineárně vynásobený obraz, který se uloží do proměnné *imageX*, kde *X* slouží k označení, o který obraz jde. Nově vytvořený obrázek se poté zobrazí ve střední části apletu v příslušném sloupci. Stejný postup probíhá i v případě, že je zvolena konvexní kombinace.

Konec třídy obsahuje funkci paint(), která slouží pro vykreslování obrázků a jiných grafických prvků v apletu.



Obr. 2.12: Prostředí apletu pro lineární kombinaci obrazů.

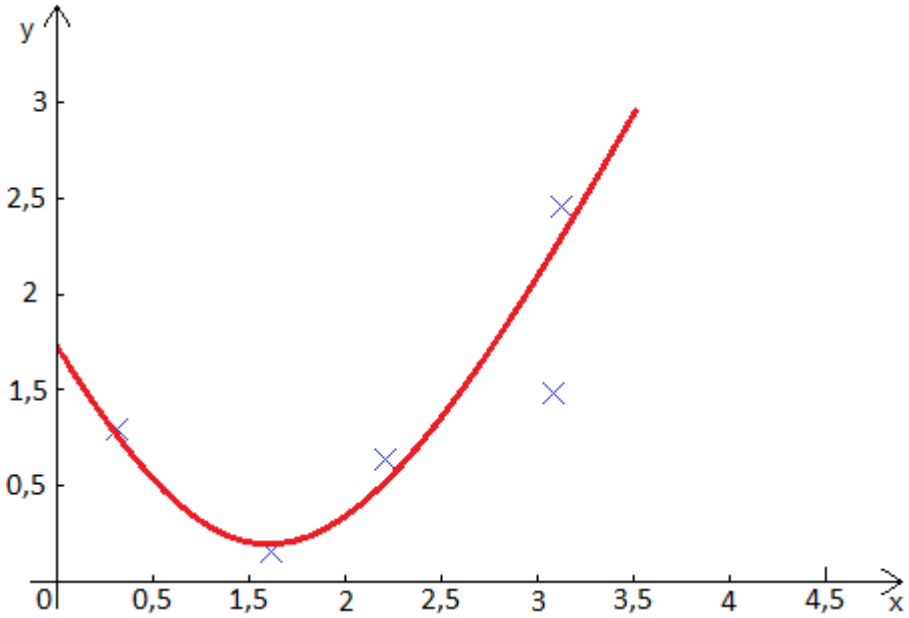
3 METODA NEJMENŠÍCH ČTVERCŮ

3.1 Teoretická část

Abychom mohli zcela porozumět, jak metoda nejmenších čtverců vlastně funguje, potřebujeme si přiblížit základní pojmy, jako je aproximace, nebo lineární regrese.

3.1.1 Aproximace

Samotné slovo aproximace můžeme do češtiny přeložit jako přiblížení, nebo by se dalo říci i zaokrouhlení. V matematice jde o prokládání nepřesných (naměřených) hodnot funkcí, která je pro tyto hodnoty vhodná. V souvislosti s aproximací se v matematice můžeme setkat se symboly jako např. ≈, kterými označujeme přibližný (aproximovaný) výsledek. Aproximace se může také využívat např. při opakovaných měřeních, kdy se výsledky od sebe liší a my hledáme nejpravděpodobnější výsledek [1]. Příklad aproximace můžeme vidět na obrázku 3.1.



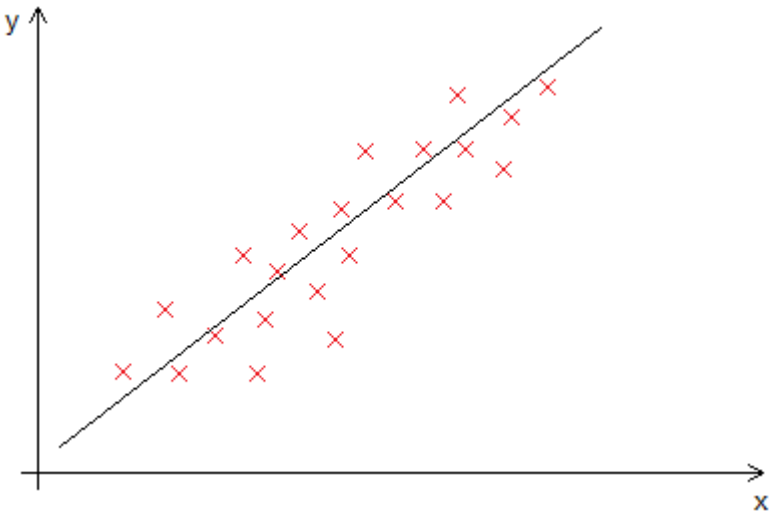
Obr. 3.1: Příklad aproximace několika bodů pomocí paraboly.

3.1.2 Lineární regrese

Ve statistice se místo názvu metoda nejmenších čtverců vžil název regrese. Jelikož obecná nelineární regrese by se těžko počítala, omezujeme se proto velmi často na regresi lineární. Jedná se o proložení souboru bodů v grafu funkcí, přičemž o bodech

v grafu platí, že jejich souřadnice na ose x jsou přesné, kdežto souřadnice na ose y vykazují zatížení chybou. Přitom předpokládáme, že body v grafu lze vyjádřit vybranou funkcí $x = \sum_i \beta_i \cdot f_i(x)$. V této rovnici označuje x bod v grafu, β_i koeficient, kterým násobíme, a $f_i(x)$ je funkce, kterou aproximujeme.

Lineární regrese v podstatě hledá takovou funkci, u které je součet druhých mocnin odchylek souřadnic y co nejmenší [1]. Příklad můžeme vidět na obrázku 3.2.



Obr. 3.2: Lineární regrese.

3.1.3 Aproximace metodou nejmenších čtverců

Metoda nejmenších čtverců je speciální případ aproximace, kdy k aproximaci předem daných bodů v rovině použijeme některou z elementárních funkcí, jako např. přímku, parabolu, nebo obecný polynom s předem daným stupněm. Snažíme se tak zjistit nejpravděpodobnější průběh této funkce.

Formulace příkladu by mohla znít například následovně:

Je dána množina bodů $[x_i, y_i]$, *kde* $i = 0, \ldots, n$; *a zároveň platí, že* $n \geq 1$. *Víme, že hodnoty* y_i *nejsou přesné. Úkolem je najít funkci* $y = f(x, \beta)$, *závislou na parametru* β, *která co nejlépe vystihuje skutečnou závislost proměnné* y *na* x [3].

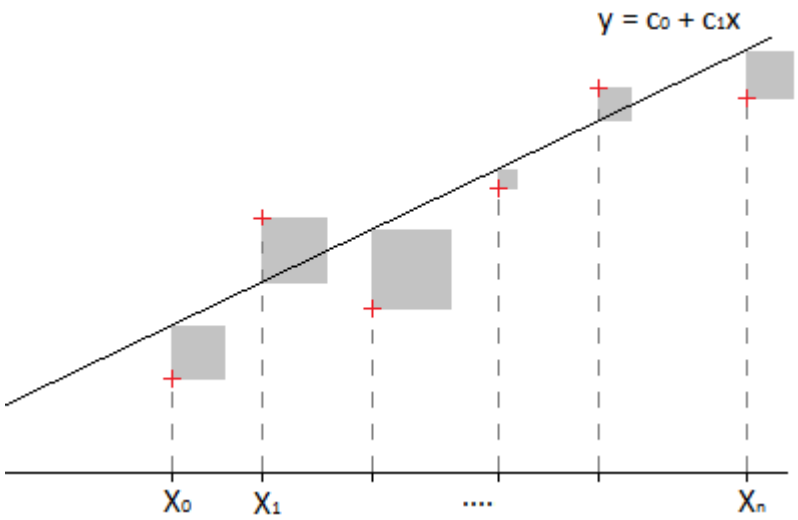
Jak již samotný název napovídá, cílem této metody je získat takové řešení, které nám v součtu dá nejmenší součet druhých mocnin chyb, tj. matematicky popsáno $min_\beta ||y - f(x, \beta)||^2$. Jinými slovy hledáme co nejmenší obsah čtverců odchylek, které při řešení vznikly. Tuto metodu lze provádět dvěma způsoby: graficky (nedoporučuje se) a numericky. Grafické provedení je velmi složité a nepřesné, proto se tato možnost nevyužívá a v praxi se nejčastěji setkáme s metodou numerickou.

Vykreslením bodů do roviny můžeme rozeznat, o jakou závislost vlastně jde a jak budeme dále postupovat. Ne vždy ale vykreslení bodů zcela jasně určí další postup. Je potřeba i jistá znalost matematických nebo fyzikálních zákonů a také určitá zkušenost.

3.1.4 Aproximace pomocí přímky

Aproximace pomocí přímky patří k nejjednodušším případům aproximace pomocí metody nejmenších čtverců. Využíváme při ní obecnou rovnici přímky ve tvaru $y = c_0x + c_1$, kde c_0 a c_1 značí koeficienty přímky a x, y označují body přímky. U metody nejmenších čtverců tedy chceme, aby součet obsahů čtverců, které vzniknou jako druhé mocniny chyb v jednotlivých bodech, byl minimální. Součet těchto čtverců nazýváme kvadratická odchylka a označujeme jej ρ^2. Lze ho vypočítat podle následujícího vztahu

$$\begin{aligned}\rho^2(c_0, c_1) &= (y_0 - c_0 - c_1x_0)^2 + (y_1 - c_0 - c_1x_1)^2 + \cdots + (y_n - c_0 - c_1x_n)^2 \\ &= \sum_{i=0}^{n} (y_i - c_0 - c_1x_i)^2 . \end{aligned} \tag{3.1}$$



Obr. 3.3: Přímka, pro niž je součet obsahů čtverců co nejmenší.

Na obrázku 3.3 můžeme vidět názornou ukázku toho, jak vypadá aproximace bodů v rovině pomocí metody nejmenších čtverců s použitím přímky.

3.1.5 Aproximace pomocí paraboly

Hned po přímce se velmi často při aproximaci využívá parabola. Aproximace se řeší obdobně jako u přímky, pouze s tím rozdílem, že k zadaným bodům hledáme rovnici paraboly. Obecnou rovnici paraboly uvádí vztah

$$y = c_0 + c_1 x + c_2 x^2. \tag{3.2}$$

Vztah pro minimální kvadratickou odchylku, tedy součet nejmenších čtverců je dán vztahem

$$\rho^2 (c_0, c_1, c_2) = \sum_{i=0}^{n} \left(y_i - c_0 - c_1 x_i - c_2 {x_i}^2 \right)^2. \tag{3.3}$$

Koeficienty $c_0, c_1 a c_2$ pak lze vypočítat pomocí soustavy tří rovnic:

$$\begin{aligned} c_0(n+1) + c_1 \sum_{i=0}^{n} x_i + c_2 \sum_{i=0}^{n} {x_i}^2 &= \sum_{i=0}^{n} y_i, \\ c_0 \sum_{i=0}^{n} x_i + c_1 \sum_{i=0}^{n} {x_i}^2 + c_2 \sum_{i=0}^{n} {x_i}^3 &= \sum_{i=0}^{n} x_i y_i, \\ c_0 \sum_{i=0}^{n} {x_i}^2 + c_1 \sum_{i=0}^{n} {x_i}^3 + c_2 \sum_{i=0}^{n} {x_i}^4 &= \sum_{i=0}^{n} {x_i}^2 y_i, \end{aligned} \tag{3.4}$$

kde $n + 1$ značí celkový počet uzlů [3].

3.1.6 Aproximace pomocí exponenciály

Další možností je při aproximaci použít exponenciální křivku s obecnou rovnicí

$$y = a e^{bx}. \tag{3.5}$$

U exponenciály lze využít stejný postup jako u přímky, nebo paraboly. Brzy ale zjistíme, že bychom k řešení dostali soustavu nelineárních rovnic o dvou neznámých, což je ovšem o poznání složitější, než řešení soustavy dvou lineárních rovnic.

Řešení tedy spočívá v převedení aproximace pro případ přímky, kdy rovnici

$$y = a e^{bx} \tag{3.6}$$

zlogaritmujeme a dostaneme vztah

$$\ln y = \ln a + bx. \tag{3.7}$$

Následně je potřeba provést substituci $c_0 = \ln a, c_1 = b$ a nyní už můžeme použít stejný postup jako při aproximaci pomocí přímky, jen s tím rozdílem, že místo y_i budeme nyní pracovat s $\ln y_i$.

3.1.7 Aproximace pomocí polynomu n-tého stupně

Aproximace pomocí přímky a paraboly můžeme také označit, jako aproximace polynomem prvního, resp. druhého stupně. Obecně můžeme podobným způsobem řešit úlohy s aproximací polynomem libovolného $m-$tého stupně. Postupujeme stejným způsobem jako u přímky a paraboly. Tentokrát ovšem dostaneme soustavu m rovnic o m neznámých. Hledáme tedy funkci s rovnicí

$$P_m(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_m x^m. \tag{3.8}$$

Soustava rovnic s koeficienty c_0 až c_m poté vypadá následovně:

$$\begin{aligned} c_0(n+1) + c_1 \sum_{i=0}^{n} x_i + ... + c_m \sum_{i=0}^{n} {x_i}^m &= \sum_{i=0}^{n} y_i \\ c_0 \sum_{i=0}^{n} x_i + c_1 \sum_{i=0}^{n} {x_i}^2 + \cdots + c_m \sum_{i=0}^{n} {x_i}^{m+1} &= \sum_{i=0}^{n} x_i y_i \\ &\vdots \\ c_0 \sum_{i=0}^{n} {x_i}^m + c_1 \sum_{i=0}^{n} {x_i}^{m+1} + \cdots + c_m \sum_{i=0}^{n} {x_i}^{2m} &= \sum_{i=0}^{n} {x_i}^m y_i \end{aligned} \tag{3.9}$$

V apletu Nejmenších čtverců je z technických důvodů možné zvolit nejvýše polynom 6. stupně. Při použitích vyšších stupňů by již výpočet trval nepřijatelně dlouhou dobu.

3.1.8 Aproximace kombinací funkcí sinus a cosinus

Poslední možností aproximace metodou nejmenších čtverců, která je v apletu použita, je aproximace za použití kombinací funkcí sinus a cosinus. Výsledná funkce má tedy tvar

$$y = a \cdot \sin(x) + b \cdot \cos(x) + c. \tag{3.10}$$

Pro vyšší stupně funkce má rovnice tvar

$$y = a \cdot \sin(2 \cdot x) + b \cdot \cos(2 \cdot x) + c \cdot \sin(x) + d \cdot \cos(x) + e, \tag{3.11}$$

$$y = a \cdot \sin(3 \cdot x) + b \cdot \cos(3 \cdot x) + c \cdot \sin(2 \cdot x) + d \cdot \cos(2 \cdot x) + e \cdot \sin(x) + f \cdot \cos(x) + g, \tag{3.12}$$

a podobně pro další vyšší stupně.

Výpočet koeficientů opět probíhá stejně jako u přímky nebo paraboly. Rozdíl je pouze v tom, že namísto hodnot $x^0, x^2, \ldots, x^n$ tvoří rovnice v soustavě rovnic

vzájemné násobky hodnot sinus(x) a cosinus(x). Soustava rovnic má tedy následující tvar:

$$\begin{aligned} c_0(n+1) + c_1 \sum_{i=0}^{n} \sin(x) + c_2 \sum_{i=0}^{n} \cos(x) &= \sum_{i=0}^{n} y_i, \\ c_0 \sum_{i=0}^{n} \sin(x_i) + c_1 \sum_{i=0}^{n} \cos(x_i) + c_2 \sum_{i=0}^{n} \sin(x_i)\cos(x_i) &= \sum_{i=0}^{n} \sin(x_i) y_i, \\ c_0 \sum_{i=0}^{n} \cos(x_i) + c_1 \sum_{i=0}^{n} \sin(x_i)\cos(x_i) + c_2 \sum_{i=0}^{n} \sin^2(x_i) &= \sum_{i=0}^{n} \sin(x_i)\cos(x_i) y_i. \end{aligned} \tag{3.13}$$

Pro funkce s vyššími stupni je postup obdobný.

3.2 Popis rozhraní apletu

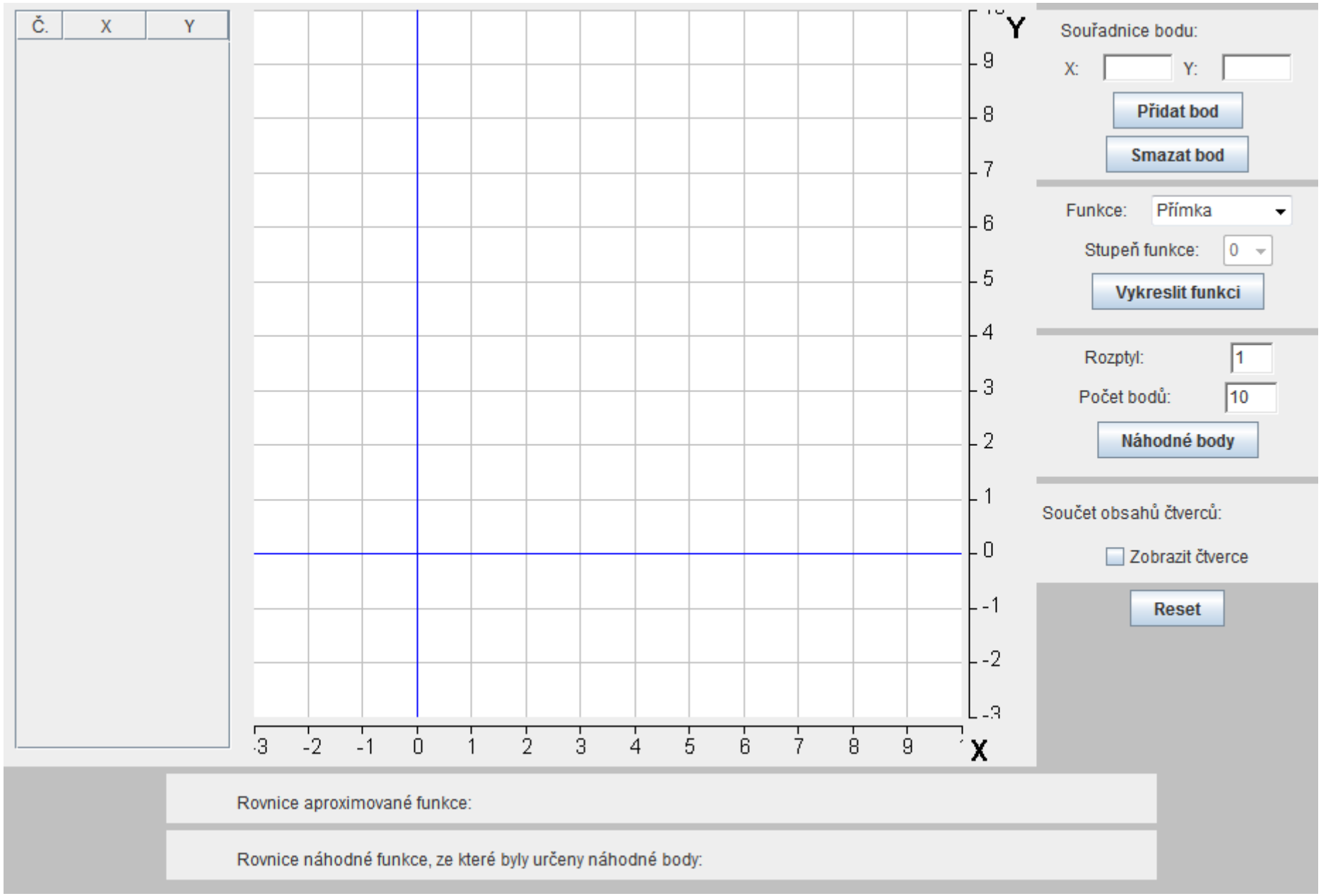
Aplet nejmenších čtverců se zabývá výpočtem aproximační funkce k proložení bodů v rovině. Na obrázku 3.4 je zobrazeno rozhraní apletu se všemi jeho prvky.

V levé části okna se nachází tabulka, která je z počátku prázdná a naplňuje se body, které uživatel do apletu zadá. Souřadnice bodů v tabulce lze libovolně upravovat nebo mazat. Hlavní část apletu se nachází uprostřed okna. Je to plátno, do kterého se zakreslují uživatelem zadané body a především se zde vykreslují i výsledné vypočítané funkce. V pravé části okna je umístěn panel s veškerými ovládacími prvky apletu. Více o těchto prvcích je uvedeno níže v části Ovládání apletu. Konečně ve spodní části apletu se nachází dvě pole, z nichž to horní, pro nás nejdůležitější, slouží k výpisu výsledné rovnice funkce, jíž aproximujeme zadané body.

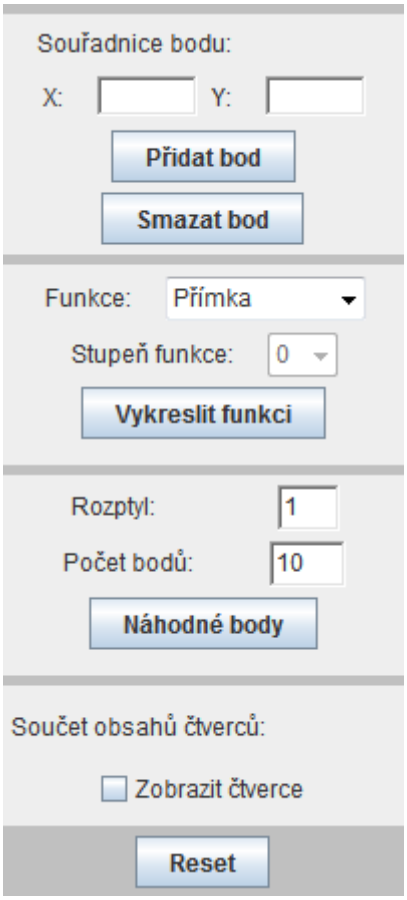
3.2.1 Ovládání apletu

Aplet je vytvořen tak, aby jeho ovládání bylo, pokud možno, pro uživatele přehledné a co nejjednodušší. Celý aplet se ovládá pouze pomocí ovládacích prvků, které se nachází v pravé části samotného apletu. K nalezení jsou zde celkem čtyři panely s různými prvky, kdy každý z nich má své specifické využití. Na obrázku 3.5 lze vidět panely s ovládacími prvky.

Panel, který se nachází nejvýše, slouží k přidávání jednotlivých bodů do tabulky. Obsahuje dvě textová pole označená X a Y, do kterých uživatel zadává souřadnice bodu. Souřadnice mohou mít libovolnou velikost a mohou být i záporné. Při zadávání desetinných hodnot je potřeba jako oddělovač použít tečku. Čárka bude brána jako chybný znak. Panel dále obsahuje dvě tlačítka. První, s označením *Přidat bod*, přidá souřadnice z obou textových polí do tabulky a vykreslí bod na plátno. Druhé tlačítko slouží k mazání bodů v tabulce. Stačí pouze v tabulce označit bod, který chce uživatel smazat a poté kliknout na tlačítko *Smazat bod*. Vybraný bod poté zmizí z tabulky i z plátna.



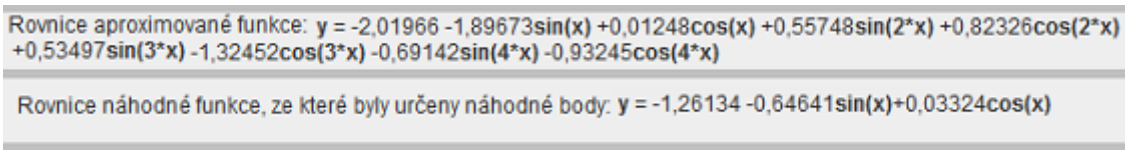
Obr. 3.4: Prostředí apletu metody nejmenších čtverců.



Obr. 3.5: Ovládací prvky apletu.

Druhý panel nabízí uživateli možnost výběru funkce, kterou chce zadané body aproximovat. Funkci lze zvolit kliknutím na rolovací lištu a výběrem z nabídky funkcí. Na výběr jsou následující funkce: přímka, parabola, exponenciála, obecný polynom $n-$tého stupně, kombinace funkcí sinus a cosinus. V případě zvolení polynomu, nebo kombinace funkcí sinus, cosinus se uživateli zpřístupní možnost volby stupně zvolené funkce. Lze tak aproximovat polynomem nultého až šestého stupně (vyšší stupně už jsou příliš časově náročné pro výpočet, proto nejsou v apletu použity). Po zvolení funkce, a případně jejího stupně, stačí stisknout tlačítko *Vykreslit funkci*. Následně proběhne výpočet, jehož výsledkem bude funkce, která nejlépe splňuje požadavky na funkci k aproximaci zadaných bodů metodou nejmenších čtverců. Rovnice výsledné funkce se zobrazí ve spodní části okna a do plátna se vykreslí její průběh.

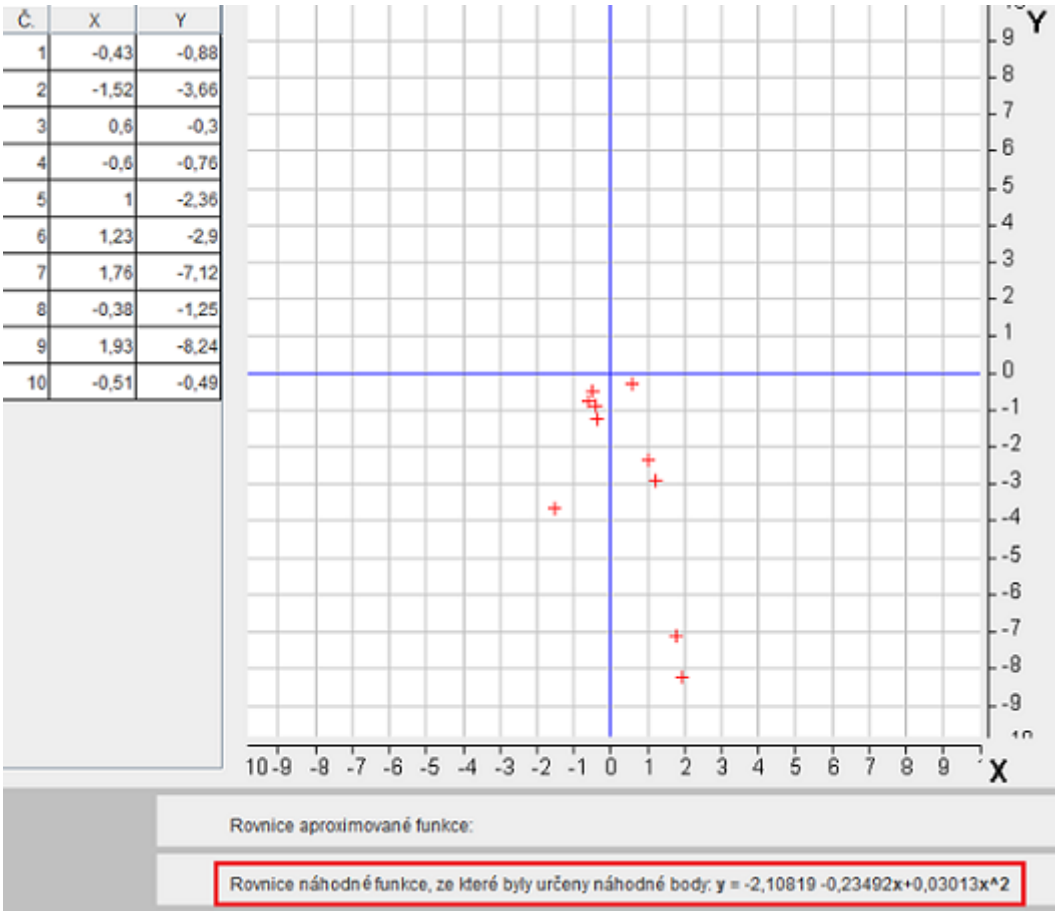
Třetí panel shora byl vytvořen ke zjednodušení a urychlení procesu zadávání bodů uživatelem. Funguje na principu přidávání náhodných bodů do tabulky a plátna s parametry, které uživatel předem zadá. Celá funkce spočívá v tom, že aplet vytvoří funkci se zcela náhodnými koeficienty a pro tuto funkci vybere předem daný počet bodů, ke kterým se přidá gaussovský šum se zvolenými parametry. Těmito body poté zaplní tabulku a zakreslí je do plátna. Parametry náhodné funkce jsou nahodilé, podle Gaussova rozdělení. Uživatel pouze zvolí, která funkce to bude (z nabídky ve druhém panelu) a do textových polí zadá potřebné parametry. Dvě textová pole slouží k zadávání právě těchto parametrů. První pole určuje odchylku, resp. vzdálenost, vytvořených bodů od náhodně vytvořené funkce. Pokud uživatel zadá hodnotu 0, budou se všechny body nacházet právě na dané funkci. Druhé textové pole určuje počet vytvořených bodů. Je třeba však mít na paměti, že u všech funkcí je potřeba mít vytvořeny alespoň 2 body (u některých funkcí, jako např. u paraboly, alespoň 3 body), aby aproximace mohla proběhnout. Pokud již uživatel má zadané tyto dvě hodnoty, stačí pouze kliknout na tlačítko *Náhodné body* a body se automaticky vytvoří. Dále již zbývá jen vypočítat funkci, kterou budeme aproximovat a vykreslit ji do plátna. K tomu využijeme tlačítko *Vykreslit funkci* ze druhého panelu.



Obr. 3.6: Rovnice výsledné funkce a funkce náhodně generovaných bodů z obrázku 3.9.

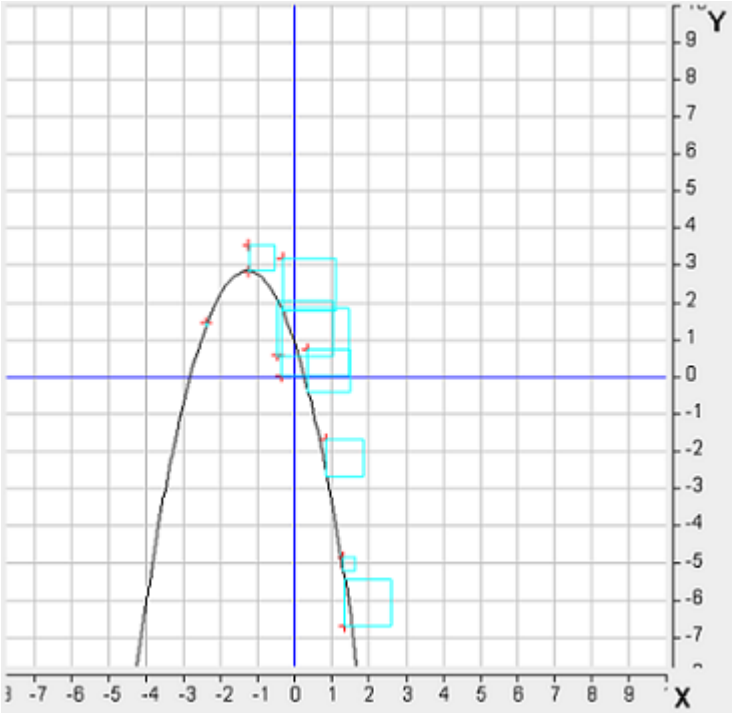
Na obrázku 3.7 můžeme vidět příklad náhodně vytvořených bodů, které vznikly z původní náhodné funkce, jejíž rovnici lze vidět ve spodní části.

Poslední panel obsahuje pouze dva prvky. Prvním z nich je textová položka, která po výpočtu aproximované funkce zobrazí vypočtené hodnoty reziduí, tedy součty obsahů čtverců, které vzniknou odchylkami bodů od funkce. Druhou položkou je možnost zobrazení čtverců, které vznikly odchylkami. Uživatel si může sám zvolit, zda chce tyto čtverce v plátně zobrazit, nebo nikoliv. Obrázek 3.8 ukazuje, jak v praxi vypadá výsledné zobrazení vypočtené funkce i se zobrazenými čtverci.

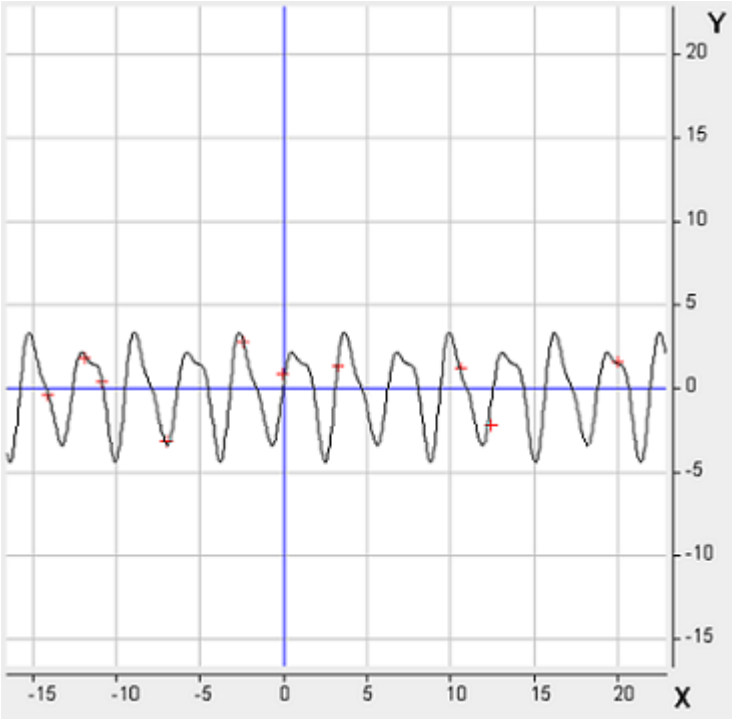


Obr. 3.7: Příklad náhodně vytvořených bodů.

Jako poslední ovládací prvek se pod všemi panely nachází tlačítko *Reset*. Je záměrně umístěno mimo ostatní ovládací prvky z důvodu jeho funkce. Stiskem tohoto tlačítka dojde k resetování apletu a k jeho uvedení do počátečního stavu, tedy do stavu hned po spuštění. Veškeré body z tabulky i z plátna budou vymazány a všechny parametry budou nastaveny na původní hodnoty. Aby nedošlo k nechtěnému resetu při náhodném kliknutí na tlačítko, je tato funkce opatřena ochrannou pojistkou. Po kliknutí na tlačítko se objeví varovné okénko s možností volby potvrdit a provést reset, nebo příkaz zrušit a neprovádět žádné změny.



Obr. 3.8: Příklad náhodně vytvořených bodů, již proložených aproximovanou funkcí, se zobrazenými čtverci odchylek.



Obr. 3.9: Příklad výstupu apletu při volbě kombinací funkcí sinus a cosinus.

3.3 Popis funkcí ve zdrojovém kódu

Stejně jako tomu je u prvního apletu i tento aplet je rozšířen o metodu *JApplet*, z důvodu funkčnosti některých komponent a hlavně z důvodu použití v internetovém prohlížeči. Aplet se nachází v balíku s názvem projekt2, jenž je součástí projektu s názvem *Nejmensi ctverce*.

Veškeré funkce apletu se nachází v hlavní třídě s názvem *Projekt2()*, která tvoří tělo apletu.

V úvodu této třídy se nachází metoda *main()*, která obsahuje pouze parametry okna, v němž se aplet nachází. Pro funkci apletu však neobsahuje příliš důležité informace.

Dále tato třída obsahuje inicializaci všech globálních proměnných a veškerých ovládacích prvků apletu, včetně všech popisků, tlačítek a textových polí. Následuje funkce s názvem *TableModel*, která vytváří tabulku a nastavuje veškeré její parametry jako např. počet sloupců, maximální počet řádků, možnost editovat buňky, apod.. Jako další se zde nachází dva prvky pro grafické znázornění hodnot souřadnic na plátně. Za nimi již je část věnovaná samotnému grafickému plátnu. Mimo parametrů samotného plátna obsahuje především důležité algoritmy pro vykreslování průběhů jednotlivých funkcí, a následně i funkci pro zobrazování čtverců vzniklých odchylkami.

Po tomto bloku následuje inicializace použitých tlačítek. Jako první jsou popsány funkce tlačítka s označením *reset*. Toto tlačítko slouží k uvedení apletu do původního stavu. Obsahuje tedy nastavení všech globálních proměnných na jejich úvodní hodnoty a rovněž veškeré popisky se vrací do původního tvaru. Také dochází ke smazání veškerých hodnot z tabulky a promazání plátna.

Pod tlačítkem s označením *plot* se nachází veškeré prováděné výpočty, které vedou k výpočtu koeficientů hledaných funkcí a tím pádem k získání výsledné funkce. Pro každou funkci je zvolen vlastní algoritmus, který vede k výpočtu koeficientů funkce.

Tlačítko *random* pomocí parametrů, zadaných do příslušných polí, vytvoří náhodné funkce, podle kterých pak do plátna vykreslí náhodné body. Slouží pro jednodušší zadávání bodů do tabulky.

Jako poslední funkční prvek se v kódu nachází tlačítko *delete*, které vymaže označený řádek tabulky a následujícím řádkům sníží indexovou hodnotu o jedničku a posune je v tabulce o jedno místo nahoru.

Závěr kódu tvoří především příkazy pro nastavení rozměrů a umístění veškerých prvků apletu. Celý aplet je rozdělen do panelů, které jsou rozmístěny do využitého prostoru. Popis jednotlivých panelů je detailně rozebrán v části 3.2.

Na úplném konci je umístěna funkce pro výpočet determinantu matic, kterého

se využívá při výpočtech. Výpočet determinantu probíhá na principu rekurzivní expanze řádek po řádku. Matice o velikosti $N \times N$ je v podstatě rozdělena do N matic o velikosti $(N-1) \times (N-1)$ a tyto matice jsou pak stejným způsobem rozděleny na menší matice. Tento proces končí, pokud matice má pouze jeden prvek, tedy $N = 1$. Pomocí sčítání a násobení určitých prvků každé z matic dostaneme na závěr požadovanou hodnotu determinantu původní matice.

4 HÁZENÍ ŠIPKAMI NA TERČ

Třetí část bakalářské práce se zabývá tvorbou apletu pro doporučení mířicího bodu při házení šipkami na terč pomocí dvojrozměrné konvoluce a Gaussova rozdělení. Tento aplet poskytuje uživateli informace o tom, na jaké místo terče by měl zamířit, pokud chce hodem průměrně získat co největší počet bodů. K těmto výpočtům je v apletu využito normálního rozdělení pravděpodobnosti a dvojrozměrné konvoluce.

4.1 Teoretická část

4.1.1 Konvoluce

Princip konvoluce spočívá ve vytvoření nové funkce ze dvou předem známých funkcí. Vztah pro výpočet spojité konvoluce funkcí f(x) a g(x)je následovný:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(\alpha)g(x-\alpha)\mathrm{d}\alpha. \tag{4.1}$$

Pro potřeby využití, např. v počítačové grafice, je potřeba konvoluci diskretizovat a vytvořit tak diskrétní konvoluci se vztahem:

$$I(x,y) = \sum_{i=-a}^{a} \sum_{j=-a}^{a} f(x-i, y-j) \cdot h(i,j), \tag{4.2}$$

kde $I(x,y)$ je intenzita výsledného pixelu o souřadnicích $[x,y]$, $f(x,y)$ značí intenzitu vstupního obrazu na souřadnicích $[x,y]$ a $h(i,j)$ zastupuje intenzitu bodu o souřadnicích $[i,j]$ v masce [10].

V apletu bylo nutné diskretizovat plochu terče i dvojrozměrnou Gaussovu funkci, ze které poté vznikla matice s hodnotami hustot pravděpodobností. Postup diskretizace je více popsán v částech 4.3.1 a 4.3.2.

V našem případě slouží konvoluce k získání bodu s největší hustotou pravděpodobnosti. Toho je dosaženo tím, že vzájemně násobíme prvky dvou matic, jejichž prvky předem známe, nebo jsme schopni je dopočítat. Pokud je jedna z matic tvořena méně prvky, než matice druhá, umístíme ji na začátek větší matice a postupně ji po ní posouváme, dokud se nedostaneme až na její konec. Vynásobené hodnoty prvků obou matic poté sečteme a dostáváme výsledný součet. Výsledné součty, vzniklé při každém posunu menší matice, na závěr porovnáme a zjistíme, ve které pozici menší matice byl součet největší.

4.1.2 Normální rozdělení

Normální rozdělení pravděpodobnosti, také označováno jako Gaussovo rozdělení pravděpodobnosti, je nejvýznamnější ze všech pravděpodobnostních rozdělení [3].

Hustota tohoto rozdělení má tvar tzv. Gaussovy křivky a můžeme jím popsat ty veličiny, jejichž hodnoty jsou nějak ovlivněny velkým množstvím faktorů. Hustotu náhodné veličiny X s normálním rozdělením, s rozptylem $\mathrm{D}X = \sigma^2, (\sigma > 0)$ a střední hodnotou $\mathrm{E}X = \mu, (\mu \in \mathbb{R})$ lze vypočítat pomocí vztahu

$$f(x) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}, x \in \mathbb{R}, \tag{4.3}$$

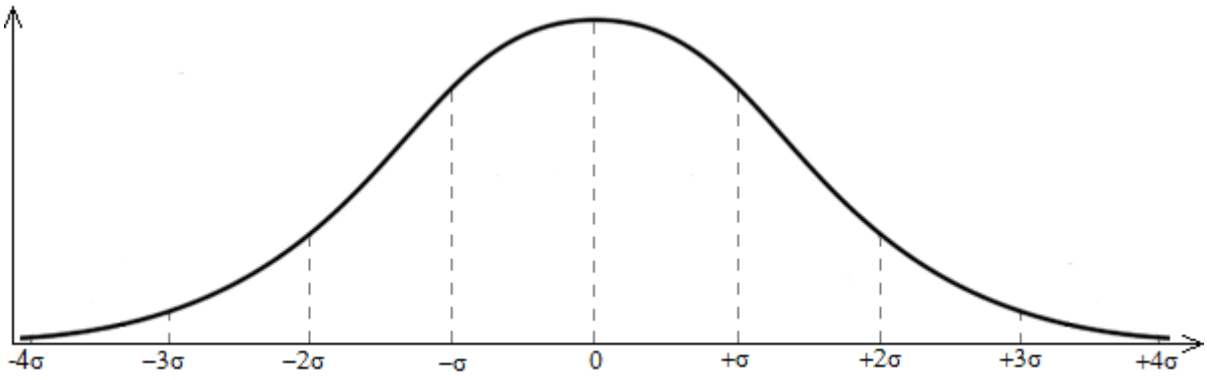
který také můžeme, podle [6], zkráceně zapsat jako

$$X \sim N(\mu, \sigma^2). \tag{4.4}$$

Pro zjednodušení výpočtů lze počítat se střední hodnotou $\mu = 0$. Vztah je tedy pro výpočet hustoty pravděpodobnosti zjednodušen do tvaru

$$f(x) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot e^{-\frac{x^2}{2\sigma^2}}, x \in \mathbb{R}. \tag{4.5}$$

Příklad znázornění hustoty pravděpodobnosti pomocí Gaussovy křivky můžeme vidět na obrázku 4.1.



Obr. 4.1: Gaussova křivka normálního rozdělení.

4.1.3 Gaussovo rozdělení ve 2D

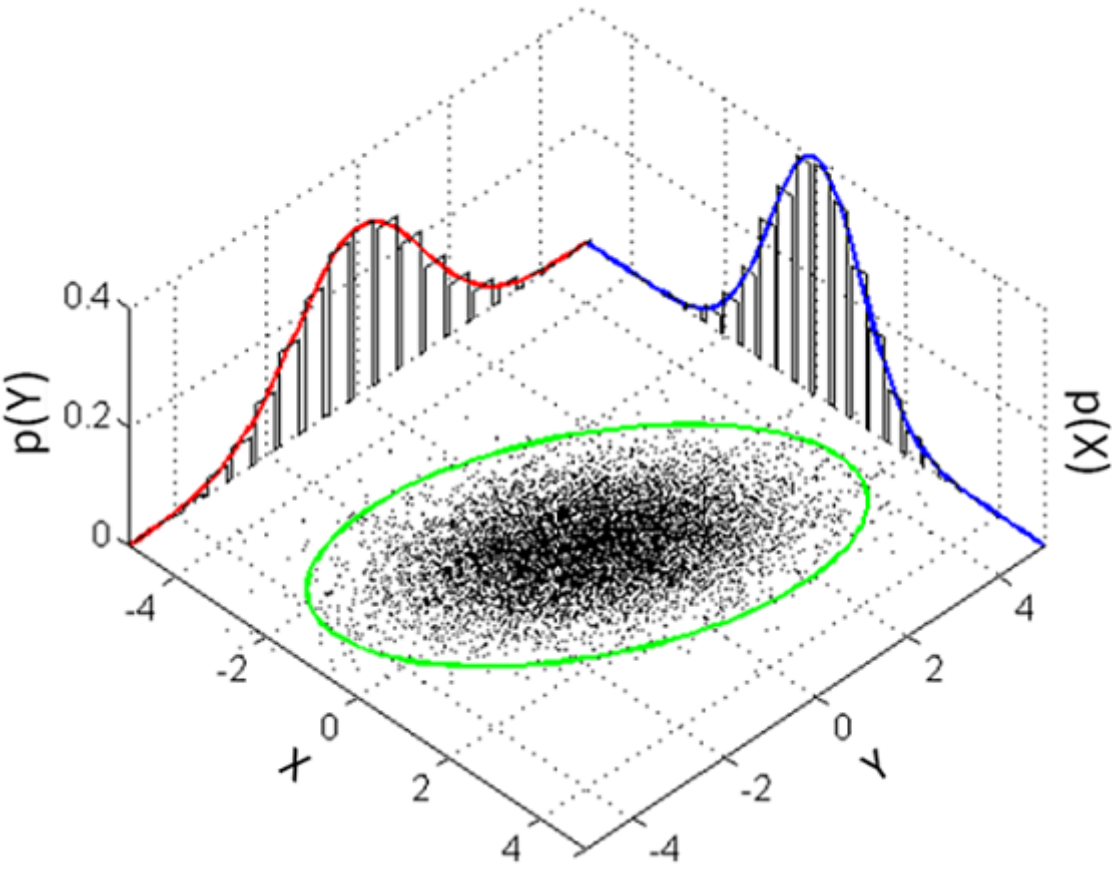
Dvourozměrné Gaussovo rozdělení funguje na stejném principu, jako jednorozměrné. Rozdíl je pouze v tom, že 2D rozdělení funguje v poli, které má dva rozměry a je tedy náročnější na výpočet hustoty pravděpodobnosti. Pro lepší představu se můžeme na obrázku 4.2 podívat, jak vypadá rozložení hustoty pravděpodobnosti ve dvourozměrném prostoru pomocí 3D modelu tohoto zobrazení. V apletu se ale omezíme pouze na rotačně symetrické zobrazení, což znamená, že průběhy obou funkcí v obou směrech jsou totožné.
Hustota obecného dvourozměrného Gaussova rozdělení je následující

$$f(x,y) = \frac{1}{2\pi \cdot \sigma_x \sigma_y \sqrt{1-\rho^2}} \cdot e^{\left[-\frac{1}{2(1-\rho^2)}\left(\frac{(x-\mu_x)^2}{\sigma_x^2}+\frac{(y-\mu_y)^2}{\sigma_y^2}-\frac{2\rho(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y}\right)\right]}, x, y \in \mathbb{R}. \tag{4.6}$$

V tomto vztahu označuje σ_x, σ_x rozptyl $\mathrm{D}X$ resp. $\mathrm{D}Y$, přičemž platí, že $\sigma_x > 0 \wedge \sigma_y > 0$, μ_x, μ_y označují střední hodnoty $\mathrm{E}X$ resp. $\mathrm{E}Y$, $(\mu_x, \mu_y \in \mathbb{R})$ a ρ je korelační koeficient mezi X a Y, $(-1 < \rho < 1)$. Jelikož budeme při tvorbě apletu využívat pouze rotační symetrické zobrazení, vztah se nám o něco zjednoduší a sice tak, že $\sigma_x = \sigma_y = \sigma$ a $\rho = 0$.

Zjednodušený tvar rovnice pro výpočet hustoty pravděpodobnosti dvojrozměrné Gaussovy funkce je pro rotační symetrické zobrazení následující:

$$f(x,y) = \frac{1}{2\pi \cdot \sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}, x, y \in \mathbb{R}. \tag{4.7}$$

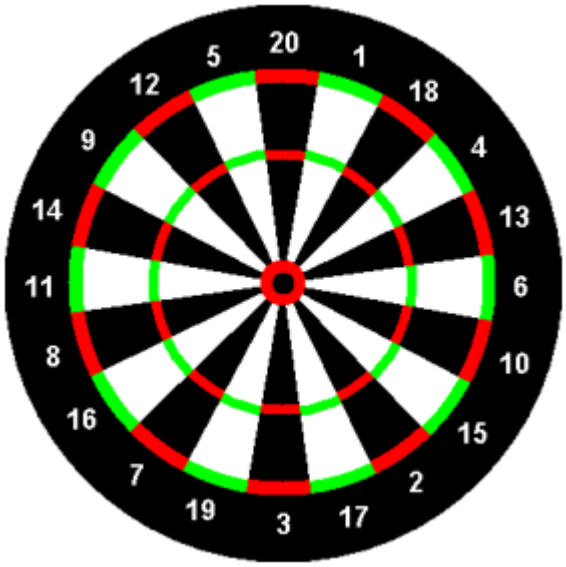


Obr. 4.2: 2D Gaussovo rozdělení [4].

4.2 Popis rozhraní apletu

Jak již bylo řečeno, aplet se zabývá výpočtem a následným doporučením mířicího bodu na šipkovém terči a to pomocí hustoty pravděpodobnosti, se kterou uživatel může trefit vybrané místo, na které pokud bude mířit, nahází průměrně největší počtu bodů. Pozice nejvýhodnějšího mířicího bodu je v apletu zaznačena přímo do obrázku terče s tím, že pozice na terči odpovídá skutečnému doporučenému místu na reálném terči.

Opět bylo při vytváření apletu důležité myslet na to, aby byl uživatelsky přívětivý, přehledný a jednoduchý na ovládání. Toho bylo dosaženo vytvořením pouze minimálního počtu ovládacích prvků.



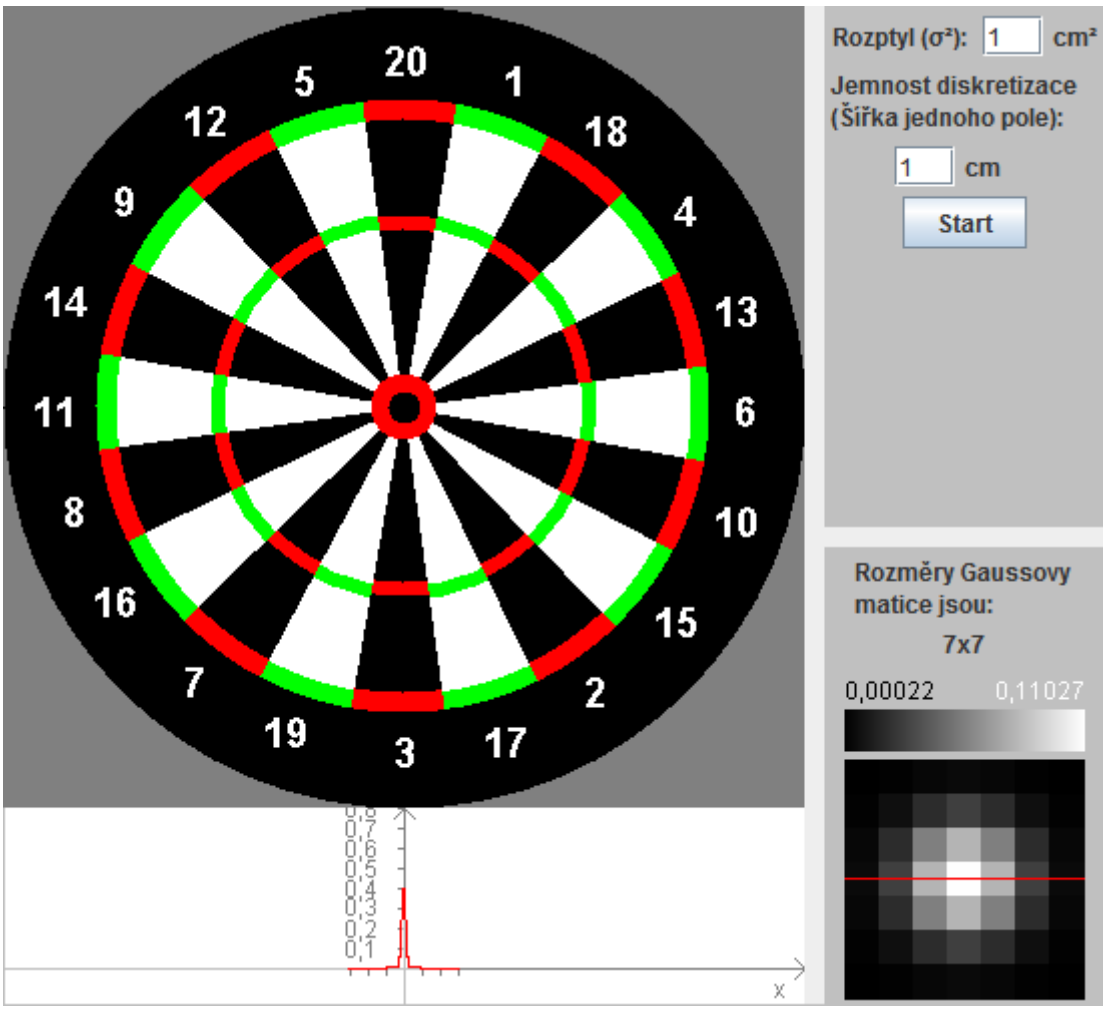
Obr. 4.3: Terč použitý v apletu.

Aplet Doporučení mířicího bodu obsahuje tři hlavní prvky:

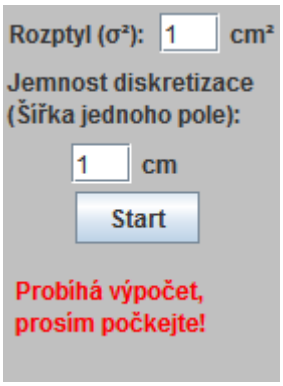
1. Hlavní panel s obrázkem terče
2. Panel s ovládacími prvky
3. Panel s grafickým znázorněním průběhu Gaussovy funkce

Rozhraní apletu a rozmístění jednotlivých bloků lze vidět na obrázku 4.4.

Hlavní panel s obrázkem terče tvoří velkou část okna apletu. Obsahuje grafické znázornění reálného terče s tím, že pro zobrazení bylo potřeba rozměry terče upravit, aby odpovídal požadavkům. Průměr vnějšího kruhu reálného terče by měl, podle pravidel, být 13 a 1/4 palce, což je v přepočtu přibližně 33,655 cm. Terč znázorněný v apletu má průměr 306 px, tudíž jeden centimetr na terči v apletu se rovná $\frac{306}{33{,}655} = 9{,}09$ px. Rozměr jednoho centimetru vyjádřeného v pixelech byl pro výpočty, které aplet provádí, klíčový. Do prostoru terče se po provedených výpočtech zobrazuje přímo doporučený bod k míření.

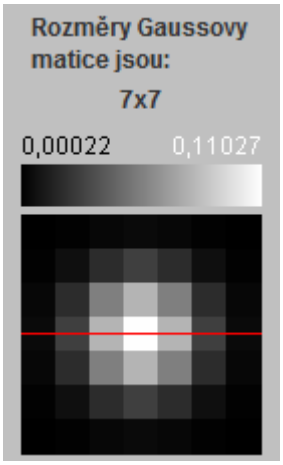


Obr. 4.4: Rozhraní apletu Doporučení mířicího bodu při hodu šipkami.



Obr. 4.5: Panel s ovládacími prvky.

Po pravé straně okna apletu se nachází panel s ovládacími prvky apletu a je jeho nedílnou součástí. Pro jednoduchost ovládání a přehlednost je celý aplet ovládán pouze jedním tlačítkem, které spouští výpočet, a dvěma textovými poli, které slouží k zadávání parametrů pro výpočty. V horní části panelu se tedy nachází textové pole pro zadávání hodnoty rozptylu σ^2. Rozptyl, který je do pole zadán je v jednotkách cm^2, se přímo vztahuje k rozměrům reálného terče. Další textové pole slouží k určení rozměrů jednotlivých buněk, do kterých je terč rozdělen. Tyto buňky určují jemnost diskretizace terče. Rozměr je zadán v centimetrech a je možné zadávat pouze hodnoty větší než 0. Jako desetinný oddělovač je u obou polí opět nutné použít tečku. Desetinnou čárku aplet vyhodnotí jako chybný znak.



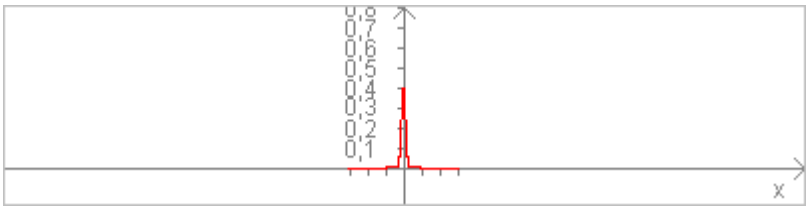
Obr. 4.6: Znázornění Gaussovy matice v apletu.

Ve spodní části pravého panelu se nachází čtverec, ve kterém je znázorněna Gaussova matice obsahující hodnoty hustot pravděpodobností. Je škálovaná tak, že nejmenší hodnoty hustoty pravděpodobnosti mají černou barvu a hodnoty nejvyšší mají barvu bílou. Čtverec je zároveň rozdělen určitého počtu menších čtverců, podle

toho, jakých rozměrů Gaussova matice nabývá. Dále je zde zobrazena škála s vypsanými krajními hodnotami hustot pravděpodobností. Nad tímto čtvercem jsou vypsány přesné rozměry aktuálně použité matice obsahující hodnoty hustot pravděpodobností.

Na závěr je ve střední části panelu vyhrazeno místo pro písemné upozornění, které se objeví v čase, kdy aplet provádí výpočty. Upozornění se zobrazuje z důvodu, aby si uživatel nemyslel, že aplet přestal pracovat a zasekl se, ale aby věděl, že aktuálně probíhá výpočet. Po skončení výpočtu a zobrazení výsledku upozornění zmizí. Panel s ovládacími prvky je znázorněn na obrázku 4.5.

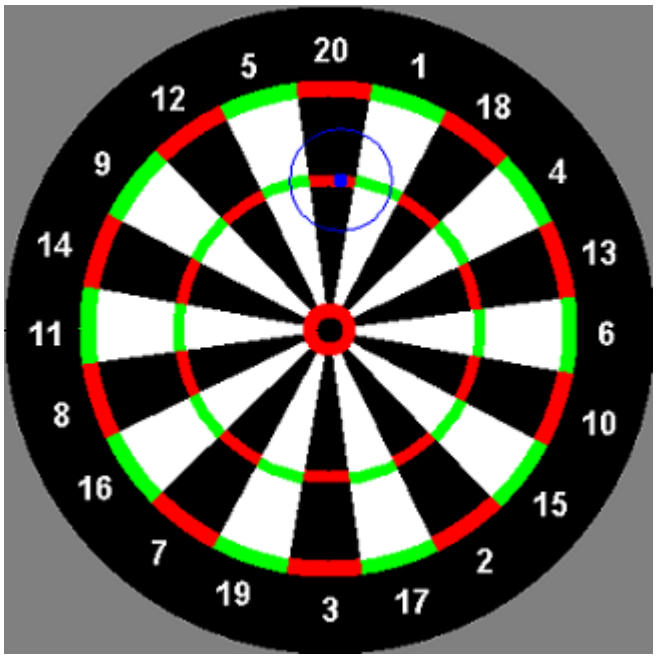
Spodní část apletu ukazuje blok, obsahující grafické znázornění řezu aktuální použité Gaussovy dvojrozměrné funkce. Řez funkce je znázorněn v grafickém ztvárnění matice s hustotami pravděpodobností, která se nachází v pravé dolní části apletu. U grafu jsou automaticky upravována měřítka vertikální osy, znázorňující hodnotu hustoty pravděpodobnosti, tak, aby byl průběh funkce vždy dobře viditelný. Horizontální osa je vztažena k velikosti samotného terče a mění se pouze pokud je změněn rozptyl σ^2. Bohužel počítačová grafika průběhy funkcí značně zkresluje a ty poté nemají příliš plynulý charakter. To je patrné převážně při zadání hodnoty rozptylu menší než 1. Tomuto problému ovšem zabránit nedokážeme.



Obr. 4.7: Plocha s grafickým znázorněním řezu Gaussovy funkce.

4.2.1 Ovládání apletu

Jak již bylo zmíněno, ovládání apletu je velmi jednoduché a přehledné. Celý aplet se ovládá pouze pomocí jediného tlačítka a dvou textových polí, které můžeme vidět na obrázku 4.5. Do horního pole uživatel zadává požadovaný rozptyl v jednotkách cm^2. Je možné zadat libovolnou hodnotu větší než 0, ovšem doporučuji zadávat hodnoty větší než 0,1 a menší než 200. Jiné hodnoty lze samozřejmě také zadávat, ovšem při kombinaci s ne příliš vhodnými hodnotami jemnosti diskretizace může dojít k velmi dlouhému výpočtu, který bude trvat i několik minut, nebo případně i ke špatné funkci apletu. Ze stejných důvodů zároveň doporučuji volit hodnoty jemnosti diskretizace v rozmezí 0,1 cm až 10 cm.



Obr. 4.8: Příklad znázornění zobrazení doporučeného mířicího bodu.

Po zadání zvolených hodnot do příslušných textových polí stačí pouze kliknout na tlačítko *Start* a aplet již sám provede výpočet a výsledek následně zobrazí na terči.

Pokud výpočet trvá nepřijatelně dlouhou dobu, doporučuji znovu načíst webovou stránku s apletem, čímž se proces ukončí a aplet se otevře v původním stavu. Při následném zadávání nových parametrů je vhodné volit již jiné hodnoty.

4.3 Popis funkcí ve zdrojovém kódu

Podobně jako u předchozích apletů, je i tento tvořen hlavní třídou *Projekt3*, která je součástí projektu *Sipky*, a ve které se nachází veškeré funkce a příkazy. Třída *Projekt3* je rozšířena o metodu *JApplet*, která umožňuje spuštění apletu a jeho provoz ve webové prohlížeči.

V úvodu této třídy se opět nachází metoda *main*, která znovu obsahuje parametry nastavení okna apletu. Bez této metody by se aplet vůbec nezobrazoval.

Následuje deklarace globálních proměnných a ovládacích prvků apletu, včetně všech popisků. Hodnoty nastavené proměnným nejsou v tuto chvíli příliš podstatné, jelikož při výpočtech se do těchto proměnných nastavují získané, nebo vypočítané hodnoty.

Po deklaraci proměnných a prvků apletu začíná důležitá část kódu, a sice blok se jménem *Display*. Tento blok slouží pro vykreslování virtuálního terče a po ukončení

výpočtu také ke znázornění doporučeného mířicího místa. Terč, jako takový, je tvořen celkem sedmi vybarvenými kruhy, které jsou poskládány na sebe od největšího po nejmenší tak, aby nejmenší byl nahoře, největší vespod, aby tak tvořily základ terče. Je důležité zachovat jejich pořadí, jinak by mohlo dojít k překrytí menšího kruhu větším. Popisky na okraji terče jsou přidány příkazem

```
g2.drawString("A", x, y);,
```

kde „A“ označuje číselnou hodnotu příslušného pole v terči a *x, y* značí souřadnice umístění popisku v prostoru bloku. V další části bloku se vybarvují jednotlivá políčka terče, podle jejich umístění. Rozmístění jednotlivých polí je v kódu umístěno až úplně na konci z toho důvodu, že se již během tvorby algoritmu pro výpočet mířicího bodu neupravuje. Podrobnější informace k této části kódu se nachází na konci této podkapitoly. V závěru bloku *Display* jsou vloženy příkazy pro vykreslení výsledného doporučeného mířicího bodu, který se na terči znázorní v podobě modrého kruhu současně s modrou kružnicí, reprezentující aktuálně zvolený rozptyl.

Pod blokem *Display* se nachází další dva grafické bloky, které slouží k názorným ukázkám průběhu Gaussovy funkce a také matice obsahující hodnoty pravděpodobnosti. Grafický blok *Graf* tvoří panel, který lze vidět ve spodní části apletu. Jedná se o grafické znázornění řezu dvojrozměrné Gaussovy funkce se zadaným rozptylem σ^2. Na horizontální ose jsou hodnoty σ, na ose vertikální jsou hodnoty hustoty pravděpodobnosti. Blok s označením *Matice* obsahuje část kódu, která v apletu vykreslí čtverec o velikosti 120×120 px a v něm znázorní aktuální podobu matice s hodnotami vypočtených hustot pravděpodobností. Pro představu o hodnotách ve čtverci je hned nad ním umístěna škála ve stupních šedi s vyznačenými krajními hodnotami. Bílá barva v tomto případě značí nejvyšší hodnotu, černá barva naopak hodnotu nejmenší.

Následuje metoda pojmenovaná po hlavní třídě *Projekt3*. Součástí této metody je ovládání tlačítka *Start*, které spouští výpočetní algoritmus, případně zobrazí chybovou hlášku, pokud byla nějaká z hodnot zadána nesprávným způsobem. Dále metoda obsahuje příkazy k uspořádání a rozmístění jednotlivých prvků do panelů, které poté tvoří samotné okno apletu. Celkem jsou zde tři panely, s tím, že panel s označením *rightPanel* je ještě rozdělen na dva další panely. Toto rozdělení bylo nutné vytvořit pro správné rozmístění komponent.

Nyní se již dostáváme k nejdůležitější části kódu. Tuto část tvoří čtyři velice důležité funkce pro výpočet doporučeného mířicího bodu:

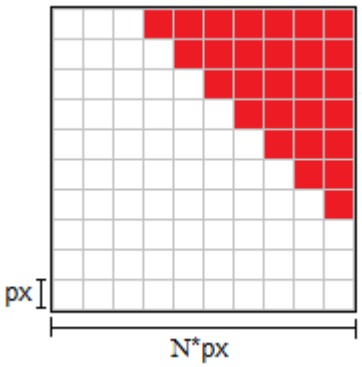
1. Funkce *matrix*
2. Funkce *matice*
3. Funkce *nastrileno*
4. Funkce *integral*

Pojmenování těchto bloků slouží pouze pro lepší orientaci v kódu.

4.3.1 Funkce *matrix*

Tato funkce slouží k diskretizaci virtuálního terče. Jelikož po uživateli při užívání apletu chceme, aby zadal, na jak drobné políčka chce celou plochu terče rozdělit, musíme vědět, jaké bodové hodnocení budou jednotlivé políčka obsahovat. Tato funkce slouží právě k účelu bodového označení všech polí, které se na terči vytvoří.

Původně bylo myšlenkou pole rozdělit na matici o velikosti $N \times N$, kde N značí počet pixelů v jednom řádku, resp. sloupci, pole. Celému poli by poté byla přiřazena hodnota, jejíž pixely by měly v poli majoritní zastoupení. Z toho by tedy plynulo, že poli na obrázku 4.9 by byla přiřazena hodnota, kterou zastupují bíle pixely, které jsou zde v největším zastoupení.



Obr. 4.9: Příklad rozložení pixelů v jednom poli terče

Toto řešení se ovšem ukázalo jako velmi nepřesné, zvláště při diskretizaci terče většími hodnotami. Proto bylo využito lepší, přesnější řešení, které je v apletu použito. Hodnoty všech pixelů pole se sečtou a tento součet se nakonec podělí celkovým počtem pixelů v poli. Dostaneme tak aritmetický průměr počet bodů, které jsou v poli zastoupeny. Tím je značně zvýšena přesnost výpočtu a proto i výsledného zobrazeného mířicího bodu.

4.3.2 Funkce *matice*

Funkce s označením *matice* je rovněž nedílnou součástí apletu a především algoritmu pro výpočet pozice mířicího bodu. Slouží k diskretizaci dvojrozměrné Gaussovy funkce. Cílem funkce je vytvoření matice o určitých rozměrech, s tím, že jednotlivé prvky matice reprezentují hustoty pravděpodobnosti dvojrozměrné Gaussovy funkce rozdělené do určitých úseků. Rozměry matice jsou vypočteny z hodnot rozptylu σ^2 a šířky pole na terči, které uživatel při práci s apletem zadává.

Hodnoty hustot pravděpodobností dvojrozměrné Gaussovy funkce lze získat například výpočtem objemu pod plochou, kterou tato funkce tvoří. Tento objem lze vypočítat dvojitým integrálem, který by ovšem v našem případě byl zbytečně příliš komplikovaný. Při tvorbě apletu jsem tedy zvolil podstatně jednodušší řešení, a sice numerický výpočet. Objem pod plochou funkce se vypočítá tak, že se sečtou objemy kvádrů, které mají velmi malou plochu základny a zároveň jsou vysoké podle funkce, pod kterou stojí. Pokud takové kvádry poskládáme vedle sebe pod celou plochu funkce a jejich objemy sečteme, dostaneme hledanou hodnotu pravděpodobnosti. Výsledek sice není zcela úplně shodný s výsledkem výpočtu pomocí dvojitého integrálu, avšak odchylka je zde zcela minimální a pro naše účely je stávající postup dostačující.

Porovnání výsledků ručně vypočítaného dvojitého integrálu s výsledkem výpočtu objemu pod plochou metodou využitou v apletu:

Ruční výpočet:

$$p = \iint \frac{1}{2\pi} \cdot e^{\left(-\frac{x^2+y^2}{2}\right)} \, \mathrm{d}x = 0,000216286. \tag{4.8}$$

Výpočet apletu:

$$p = 0,000216941895432170.$$

Protože využíváme rotačně symetrickou Gaussovu funkci, můžeme pro zkrácení doby výpočtu a urychlení celé operace vypočítat pouze jednu čtvrtinu matice, kterou poté zrcadlově překopírujeme. Vytvoříme tak polovinu matice, která se stejným způsobem opět zrcadlově překopíruje a výsledkem je matice celá. Tento postup výrazně urychluje celý proces.

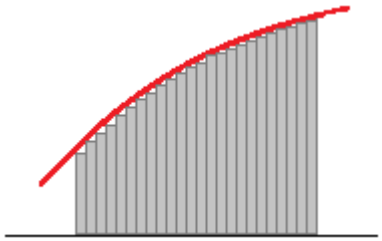
4.3.3 Funkce *nastrileno*

V této části kódu probíhají konečné výpočty a určení souřadnic výsledného bodu. Funguje na principu dvojrozměrné konvoluce. Pracuje tedy se dvěma maticemi: první je matice s hodnotami terče, která byla vytvořena ve funkci *matrix*, druhá je matice s hodnotami hustot pravděpodobností z části *matice*. Matice terče je statická a nehýbe se s ní. Naopak matice s hustotami pravděpodobností se posouvá postupně po terči a násobí svoje prvky s prvky matice terče, resp. s hodnotami, které odpovídají danému místu na terči. Tento proces začíná v levém horním rohu terče a postupně pokračuje až na úplný konec vpravo dole. Při každém posunu matice se zaznamená hodnota, která vznikla vynásobením obou matic a sečtením všech prvků výsledné matice. Pokud je tento součet větší, než dosavadní maximum, je do paměti uložena souřadnice středu pohyblivé matice. Na konci výpočtu určuje výsledné maximum souřadnice hledaného bodu, které jsou následně zakresleny přímo do terče. Pokud

dojde k tomu, že bude nalezeno více stejných maxim, určí se aritmetický průměr jejich souřadnic a ten určí výsledný hledaný bod.

4.3.4 Funkce *integral*

Funkce *integral* slouží, jak již bylo řečeno, k výpočtu objemu pod plochou dvojrozměrné Gaussovy funkce. Výsledný objem je vypočten jako součet objemů kvádrů s velmi malou plochou základny, které jsou pod plochou poskládány vedle sebe, aby zaplnily veškerý možný prostory pod plochou funkce. Na obrázku 4.10 můžeme vidět, jak je stejný postup uplatněn při výpočtu plochy pod křivkou. Při použití velmi malé základny obdélníků, resp. kvádrů, je odchylka od výsledku přesného výpočtu integrálu minimální.



Obr. 4.10: Příklad výpočtu plochy pod křivkou za použití obsahu obdélníků.

Jak již bylo zmíněno, na konci kódu apletu se nachází sekce, která slouží k přiřazení terčových hodnot příslušným pixelům. Každé políčko terče je ohraničeno vždy dvěma pomyslnými přímkami a poté dvěma kružnicemi (výjimku tvoří střed terče tvořený pouze dvěma kružnicemi). Úkolem tedy bylo získat rovnice těchto geometrických prvků a zajistit, aby pixelům, které jsou uvnitř takto ohraničeného území, byla přidělena příslušná hodnota. Algoritmu pro výpočet tedy poté stačí pouze zadat souřadnice aktuálního pixelu a obratem obdrží hodnotu, kterou pixel na terči zastupuje.

5 ZÁVĚR

Tato práce vznikla za účelem vytvoření programů pro podporu výuky kurzů BASS, BZSG a MGMP. Programy, vytvořené formou apletů, jsou vytvořeny v programovacím jazyce Java, z důvodu snadného spouštění přes webový prohlížeč. Aplety je tak možné spustit v libovolném prohlížeči pod libovolným operačním systémem.

V úvodní části práce je krátce popsán vznik a historie programovacího jazyka Java. Dále jsou zmíněny některé výhody Javy a důvody, proč jsem se rozhodl zvolit při vytváření apletů právě tento jazyk.

Druhá kapitola práce je zaměřena na problematiku lineární kombinace obrazů. V úvodní části kapitoly je zmíněna teoretická část tohoto tématu. Další část kapitoly se zabývá popisem vzhledové stránky vytvořeného apletu a je zde vysvětleno i ovládání samotného apletu a jsou přiloženy příklady, jak aplet funguje. Poslední část obsahuje popis kódu a komentuje veškeré použité funkce, které vedly k úspěšnému sestavení funkčního apletu.

Ve třetí kapitole práce je popsána teoretická část, zabývající se aproximací metodou nejmenších čtverců. V teorii je popsán samotný pojem aproximace, vysvětlení principu metody nejmenších čtverců a dále některé případy aproximace metodou nejmenších čtverců, jako je aproximace přímkou, parabolou, exponenciálou, nebo obecným polynomem n-tého stupně. Dále je zde popsána i lineární regrese. Kromě teorie, obsahuje tato část i popis postupu, který vedl k vytvoření apletu a také popis samotného apletu a jeho funkcí, včetně návodu k ovládání apletu. Kapitola také obsahuje několik obrazových příkladů výstupu výsledného apletu.

Čtvrtá část práce se zabývá teorií Gaussova rozdělení, kterého bylo využito při výpočtu a znázornění mířicího bodu na terči. Obsahuje vysvětlení pojmu normální (Gaussovo) rozdělení pro jednorozměrný i dvojrozměrný prostor, včetně matematických vztahů pro výpočet, a dále i vysvětlení pojmu konvoluce. Tato část také obsahuje popis vytvořeného apletu tak, jak ho uživatel po spuštění vidí, a také tak, jak je aplet vytvořen ve zdrojovém kódu. Jsou popsány postupy, které vedly k vytvoření potřebných funkcí pro správné fungování apletu. Rovněž je zde popsáno ovládání apletu s názornou ukázkou příkladu výstupu apletu.

LITERATURA

[1] ANDĚL, J. *Matematická statistika.* 2. vyd. Praha: Státní nakladatelství technické literatury, 1985, 346 s.

[2] HEROUT, Pavel. *Učebnice jazyka Java.* 1. vyd. České Budějovice: Kopp, 2001, 349 s. ISBN 80-723-2115-3.

[3] HLAVIČKOVÁ, Irena a Dana HLINĚNÁ. *Matematika 3.* 1. vyd. Brno: Vysoké učení technické, Fakulta elektrotechniky a komunikačních technologií, Ústav matematiky, 2007, 104 s.

[4] *Illustration of a multivariate gaussian distribution and its marginals.* In: [online]. [cit. 2014-12-05]. Dostupné z: http://upload.wikimedia.org/wikipedia/commons/8/8e/MultivariateNormal.png

[5] KISZKA, Bogdan. *1001 tipů a triků pro jazyk Java.* 1. vyd. Brno: Computer Press, 2009, 542 s. ISBN 978-80-251-2467-3.

[6] MAREK, Luboš. *Pravděpodobnost.* 1. vyd. Praha: Professional Publishing, 2012, 249 s. ISBN 9788074310874.

[7] PICHL, Karel a Jindřich PETRUCHA. *Počítačová grafika.* 2. vyd. Kunovice: Evropský polytechnický institut, 2011, 1 CD-ROM. ISBN 978-80-7314-261-2.

[8] RAJMIC, P. *Základy počítačové sazby a grafiky.* 1. Brno: FEKT VUT v Brně, 2012. 161 s. ISBN 978-80-214-4451-5.

[9] SCHILDT, Herbert. *Java 7: výukový kurz.* 1. vyd. Brno: Computer Press, 2012, 664 s. ISBN 978-80-251-3748-2.

[10] ŽÁRA, Jiří, Bedřich BENEŠ a Petr FELKEL. *Moderní počítačová grafika.* 2. vyd. Praha: Computer Press, 2005. 609 s. ISBN 80-251-0454-0.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

cm	centimetr – jednotka délky
px	Pixel - Picture element
RGB	Red, Green, Blue – označení barevných složek

A OBSAH PŘILOŽENÉHO CD

Přiložené CD obsahuje elektronickou verzi této práce ve formátu PDF, dále pak složky se soubory k jednotlivým apletům. Aplety lze snadno spustit pouhým otevřením HTML souboru v některém z webových prohlížečů.

Obsah CD je následující:

- xolber01.pdf - elektronická verze této práce
- LinearniKombinace - složka se soubory k apletu Lineární kombinace obrazů
  LinKomb.jar - balík obsahující aplet
  Projekt1.java - soubor se zdrojovým kódem apletu
  LinearniKombinace.html - HTML soubor, který otevře stránku s apletem - pouze offline
- NejmensiCtverce - složka se soubory k apletu Metoda nejmenších čtverců
  NejCtverce.jar - balík obsahující aplet
  Projekt2.java - soubor se zdrojovým kódem apletu
  NejmensiCtverce.html - HTML soubor, který otevře stránku s apletem - pouze offline
- MiriciBod - složka se soubory k apletu Doporučení mířícího bodu při hodu šipkami
  MiriciBod.jar - balík obsahující aplet
  Projekt3.java - soubor se zdrojovým kódem apletu
  Sipky.html - HTML soubor, který otevře stránku s apletem - pouze offline

Všechny tři aplety jsou dočasně umístěny na internetu a jsou volně přístupné z adresy:

- http://www.stud.feec.vutbr.cz/~xolber01/