ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra kybernetiky

Bakalářská práce

Automatické vyhodnocování obrazů z gelové elektroforézy

Vedoucí práce:

Dr. Ing. Jan Kybic

Autor:

Jiří Kaňka

Praha 2008

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne ……. ……

podpis

Poděkování

Chtěl bych především poděkovat vedoucímu mé práce Dr. Ing. Janu Kybicovi za cenné připomínky a rady při řešení problémů souvisejících s bakalářskou prací.

Dále bych zde poděkoval Ústavu živočišné fyziologie a genetiky v. v. i., České Akademie věd, Liběchov za mnoho užitečných rad ohledně gelové elektroforézy.

---VLOŽIT ZADÁNÍ BAKALÁŘSKÉ PRÁCE---

(originál nebo kopie)

Anotace

Cílem této bakalářské práce je vyvinout program určený pro zpracování obrazů z jednodimensionální gelové elektroforézy nukleových kyselin. Nejprve jsou popsány různé teoretické metody, ze kterých je možno vycházet. Dále pak algoritmus popisující chování navrženého programu, jeho ukázka, vysvětlení ovládání a přehled nejdůležitějších tříd použitých v programu.

Ke konci práce jsem se zaměřil na jeho praktickou ukázku a experimenty s programem, které ověřují jeho úspěšnost a použitelnost v praxi.

Hlavní částí této práce je elektronická příloha obsahující samotný program, jeho zdrojový kód a potřebné soubory.

Annotation

The aim of this bachelor is development of a computer program evaluating one-dimensional gel electrophoresis of nucleic acids and determining DNA fragments size. At first I described different theoretical methods used by previous authors dealing with this aim. Subsequently I described algorithm covering the behaviour of our proposed program, its sample, explanation of its control and summary of most important classes used in this program.

The last part of the bachelor I oriented on experiments with the program and practical demonstration, which verify its success and applicability in practical experience.

The main part of this bachelor represents electronic supplement which contains actual program, its source code and necessary files.

Obsah

1 Úvod

Elektroforéza nukleových kyselin je jednou z nejběžnějších metod, používaných v molekulárně genetických laboratořích (základní výzkum, medicína, ale i kriminalistika). Studium deoxyribonukleové kyseliny (DNA) totiž poskytuje obrovské množství informací, použitelné nejen v základním, ale i aplikovaném výzkumu.

K rozšíření studia DNA neobyčejně přispěl objev polymerázové řetězové reakce (PCR), která umožňuje za použití enzymu namnožit předem zvolený úsek DNA ve zkumavce (Storchová, 1998). Třebaže molekula DNA je obrovská, není ani po namnožení za běžných podmínek viditelná. Abychom ověřili výsledek jakéhokoliv pokusu či práce s DNA, musíme ji nějakým způsobem zviditelnit a zjistit její velikost. Nejjednodušší metodou, jak to provést, je právě elektroforéza.

Princip elektroforézy DNA je jednoduchý. Nukleové kyseliny jsou záporně nabité molekuly. Za jejich záporný náboj jsou zodpovědné fosfátové spojky mezi jednotlivými nukleotidy. V elektrickém poli stejnosměrného proudu se tedy vždy pohybují od katody k anodě. Aby bylo možné rozdělit jednotlivé molekuly podle velikosti, probíhá elektroforéza v gelu, který je obvykle připraven z agarózy nebo polyakrylamidu (volba gelu a jeho koncentrace závisí na tom, jak velké molekuly DNA chceme rozdělovat). Do agarózového gelu navíc musíme přidat barvu (většinou ethidiumbromid). Tato barva se vmezeří do záhybů dvoušroubovice a zůstane tam pevně navázána. Po provedení elektroforézy pak na gel posvítíme ultrafialovým světlem. Barva, vmezeřená do DNA, oranžově svítí. Vidíme tedy jednotlivé fragmenty DNA, rozdělené podle jejich velikosti.

Základní věc, kterou nám elektroforéza DNA umožňuje, je poměrně přesně určit velikost molekuly. Máme-li fragment DNA (nebo raději více fragmentů) známé velikosti, můžeme s jejich pomocí určit velikost neznámého fragmentu (molekuly) DNA. Vzhledem k tomu, že elektroforéza DNA je rutinně používanou metodou kontroly, vytvoření jednoduchého programu, který je schopen určit velikost fragmentů DNA, urychlí a standardizuje práci celé řady

laboratoří. Jisté programy, které pokrývají tuto tématiku, již existují. Vytvoření nového programu by mělo umožnit jednoduché a rychlé vyhodnocení elektroforézy, díky nezávislosti programu na platformě a možnosti přizpůsobení programu na míru případných požadavků.

2 Popis metodiky gelové elektroforézy

2.1 Popis průběhu

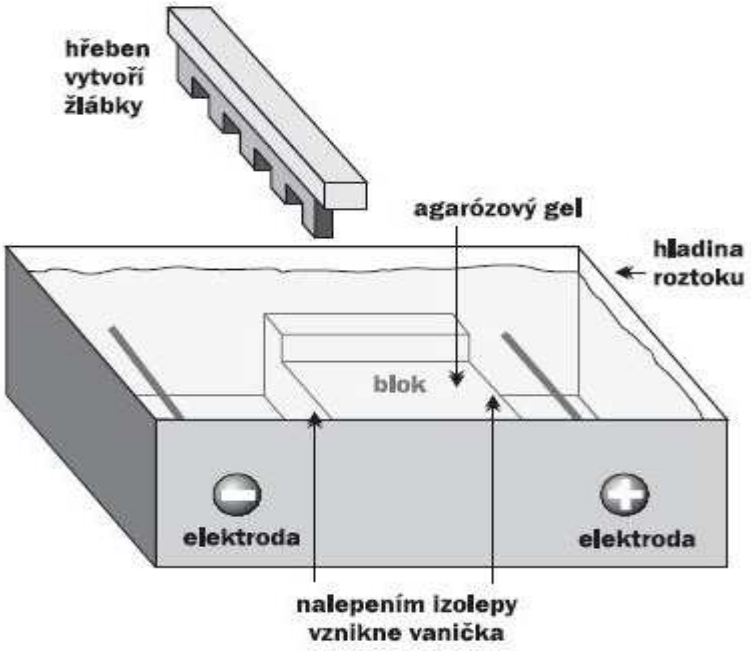
Existuje celá řada způsobů, jak provádět elektroforézu. Může probíhat v poloze horizontální nebo vertikální, lze použít různé gely, ve kterých dochází k rozdělení fragmentů (agaróza, akrylamid), rovněž složení pufru, v němž elektroforéza probíhá, může být rozdílné. (Storchová, 1998). Způsob elektroforézy volíme podle toho, jakou látku (popř. jak velké fragmenty) chceme rozdělit.

V následujícím popisu se zaměřím na typ elektroforézy nejčastěji používaný k rozdělení fragmentů DNA běžné velikosti. Jedná se o *horizontální elektroforézu v agarózovém gelu*. Nejprve je třeba rozvařit agarózu v příslušném pufru. Nejčastěji používaným pufrem je TAE (10x koncetrovaný roztok připravíme jako 0,4 M Tris-acetát, 0,01 M EDTA), koncentraci agarózy volíme podle velikosti fragmentů DNA, které chceme rozdělovat (Maniatis a kol. 1982, Tab. 2.1.).

Gel (%)	**DNA (bp)**
0,3	5 000-60 000
0,6	1 000-20 000
0,7	800-10 000
0,9	500-7 000
1,2	400-6 000
1,5	200-4 000
2,0	100-3 000

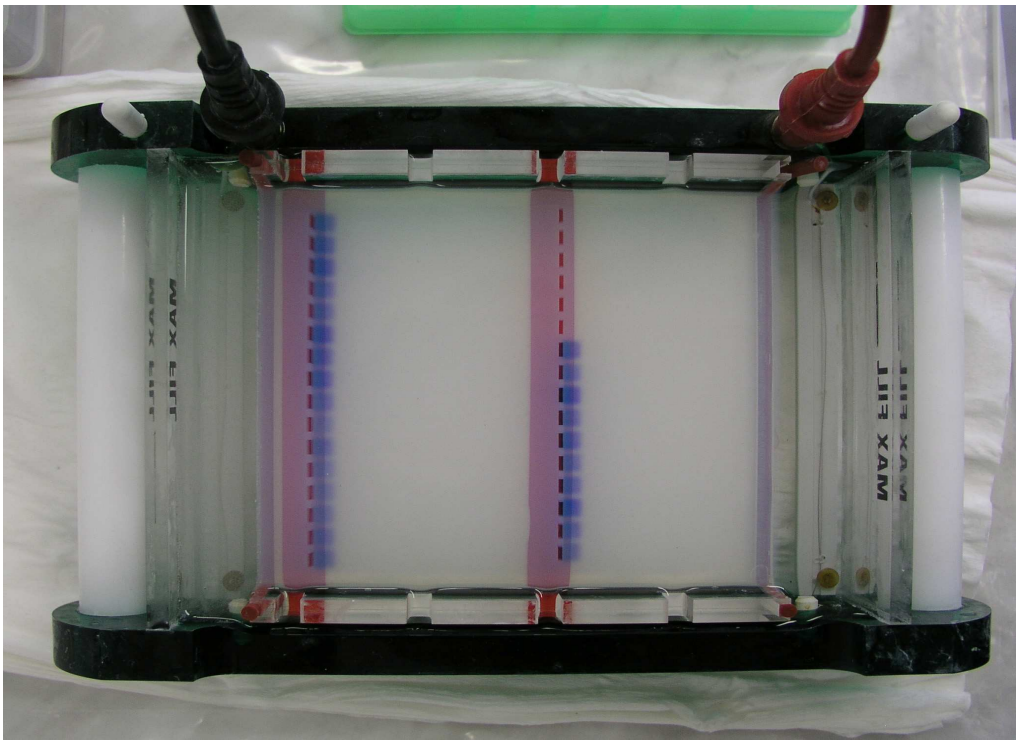
Tab. 2.1

Po zchladnutí roztoku zhruba na 60°C přidáme ethidiumbromid a roztok agarózy vlijeme do speciální formy, ve které se nachází takzvaný hřeben. Po utuhnutí agarózy (zhruba 1 hodina) a vyjmutí hřebene získáme obdélník gelu s jamkami, které slouží k nasazení vzorku DNA (Obr. 2.1).



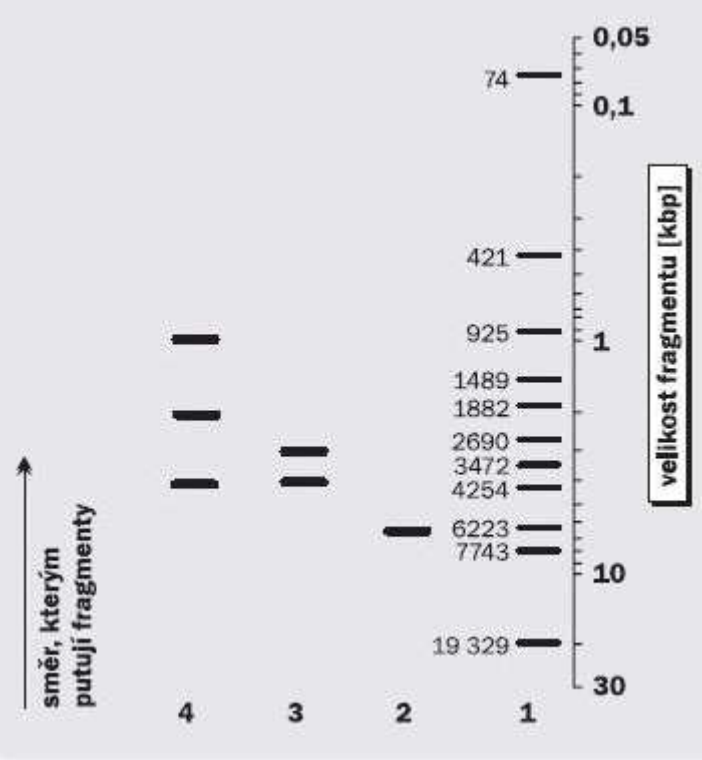
Obr. 2.1

Zkoumané vzorky DNA smícháme s nasazovacím pufrem (slouží k tomu, aby vzorek snáze klesl na dno jamky), který obsahuje modrou barvu (tato je rovněž záporně nabitá a slouží nám jako kontrola, jak rychle se záporně nabité fragmenty určité velikosti pohybují gelem) (Obr. 2.2).



Obr. 2.2

Po zapnutí stejnosměrného elektrického proudu se molekuly DNA pohybují agarózovým gelem směrem k anodě, rychlost pohybu je nepřímo úměrná velikosti fragmentu (obr. 2.3).



Obr. 2.3

V průběhu elektroforézy se DNA váže s ethidiumbromidem, přítomným v gelu (popř. i v pufru). Po skončení elektroforézy položíme obdélník gelu na prosvětlovací desku a prosvítíme ultrafialovým světlem. V tomto světle DNA, obarvená ethidiumbromidem, oranžově svítí. Obrázek gelu je dokumentován například digitálním fotoaparátem nebo CCD kamerou. Tento obraz pak můžeme použít buď k manuálnímu nebo počítačovému zpracování.

2.2 Metody vyhodnocení

Pokud chceme zjistit velikost DNA na základě vzdálenosti, kterou DNA urazila v gelu v průběhu elektroforézy, zhotovíme kalibrační graf za použití fragmentů známé velikosti (standard, „žebřík“), ve kterém vyneseme uraženou vzdálenost na gelu proti logaritmu velikosti molekuly DNA. Tento vztah je užitečný, neboť grafem je přímka, pokrývající velký rozsah velikosti fragmentů. Bohužel ve vysokomolekulární a nízkomolekulární oblasti fragmentů toto neplatí, graf se mění na nelineární křivku a při manuálním zpracování může dojít k chybě, způsobené tím, že křivka je chybně proložena body standardu (Southern, 1979; Gough a Gough, 1984; Maina a kol. 1984).

Maina a kol. (1984) popisují podmínky, které je nutné zachovat při provádění elektroforézy. Vlastní elektroforéza musí být dokonale čistá, gel musí být naléván tak, aby agaróza chladla rovnoměrně, všechny vzorky musí být ve stejném pufru a musí mít stejný objem, standard by měl obsahovat nejméně 10 fragmentů a elektroforéza by měla probíhat mezi 10 a 30 volty. Tyto podmínky však často nejsou dodrženy, zejména nízké napětí elektroforézy, neboť při něm by elektroforéza trvala neúměrně dlouho.

Southern (1979) popsal metodu pro přímý výpočet velikosti neznámých fragmentů DNA z přímého vztahu mezi velikostí fragmentu a převrácenou hodnotou uražené vzdálenosti. Rovněž u této metody je lineární oblast grafu ovlivněna podmínkami elektroforézy a je citlivá zejména na vysoké napětí a zvýšenou teplotu v průběhu elektroforézy.

Vzhledem k tomu, že manuální výpočet velikosti fragmentů je pracný a časově náročný, existuje řada prací, které se snaží o vytvoření počítačového programu, který by výsledek elektroforézy zpracovával přesněji, než to dokáže konvenční grafická metoda.

Nejprve E. Gough a N. Gough (1984) sepsali program pro kapesní programovatelný kalkulátor Hewlett-Packard, model 41C. Autoři konstatují, že i přes chyby ve výpočtech je tato metoda přesnější než manuální grafické metody. Rovněž program SPLINT byl napsán pro Hewlett-Packard 9825A (Gariepy a kol., 1986).

Další program byl sepsán v Apple Pascal a upraven pro použití na Apple II Plus (Maina a kol., 1984). Autoři konstatují, že použití programu výrazně zkrátilo čas k analýze elektroforetických dat a jsou přesvědčeni, že počítačová analýza elektroforetických dat je metodou budoucnosti.

Program ELPHOFIT je určen k analýze jednodimensionálních i dvojdimensionálních gelů a je možné ho použít na Macintoshi i IBM (Tietz, 1991).

V letech 1991-5 vznikly další programy, zpracovávající elektroforézu DNA (Yeoh, 1991; Redman, Jacobs 1991; Metzker a kol., 1995).

3 Rozbor problematiky

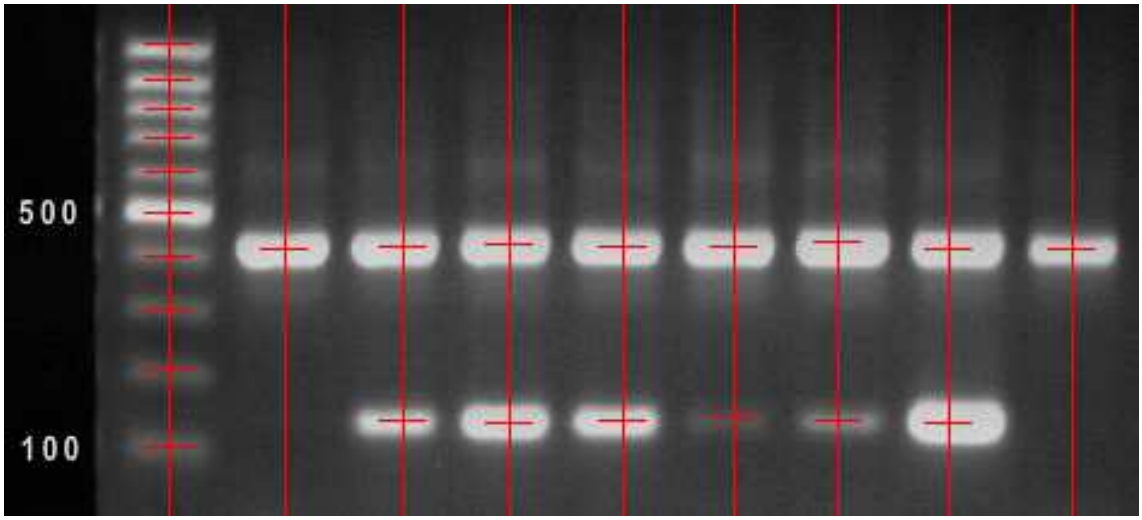
Výsledkem této bakalářské práce je samostatný, jednoduše ovladatelný program pro odečítání značek z obrazů elektroforézy. Vstupem do programu tedy jsou obrazová data. Program načítá tato data a dále je zpracovává.

Prvním úkolem je tedy otevření dat ve formátu JPG (JPEG) a jejich načtení. Dále je třeba zpracovat data na dané přímce vybrané uživatelem a určit intenzitu jednotlivých pixelů. Tyto hodnoty je třeba uložit a následně vhodným algoritmem určit jejich lokální maxima, která odpovídají hledaným značkám označujícím jednotlivé součásti směsi.

Polohu značek *markeru* (pro nás známé hodnoty jakéhosi žebříku) je potřeba porovnat s polohou pruhu (anglicky band, hledaná součást směsi) a následně podle známých velikostí částí *markeru* přepočítat na hledanou velikost *bandu*.

V programu je zapotřebí více algoritmů**,** tak, aby se pokrylo co největší množství rozdílných vstupních obrázků.

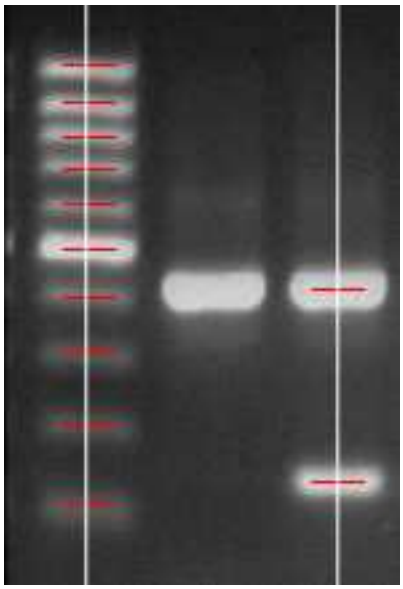
Program by měl být přenositelný mezi různými platformami. Je tudíž napsán v jazyce Java, který umožňuje snadnou přenositelnost kódu.



4 Algoritmy pro analýzu obrazu

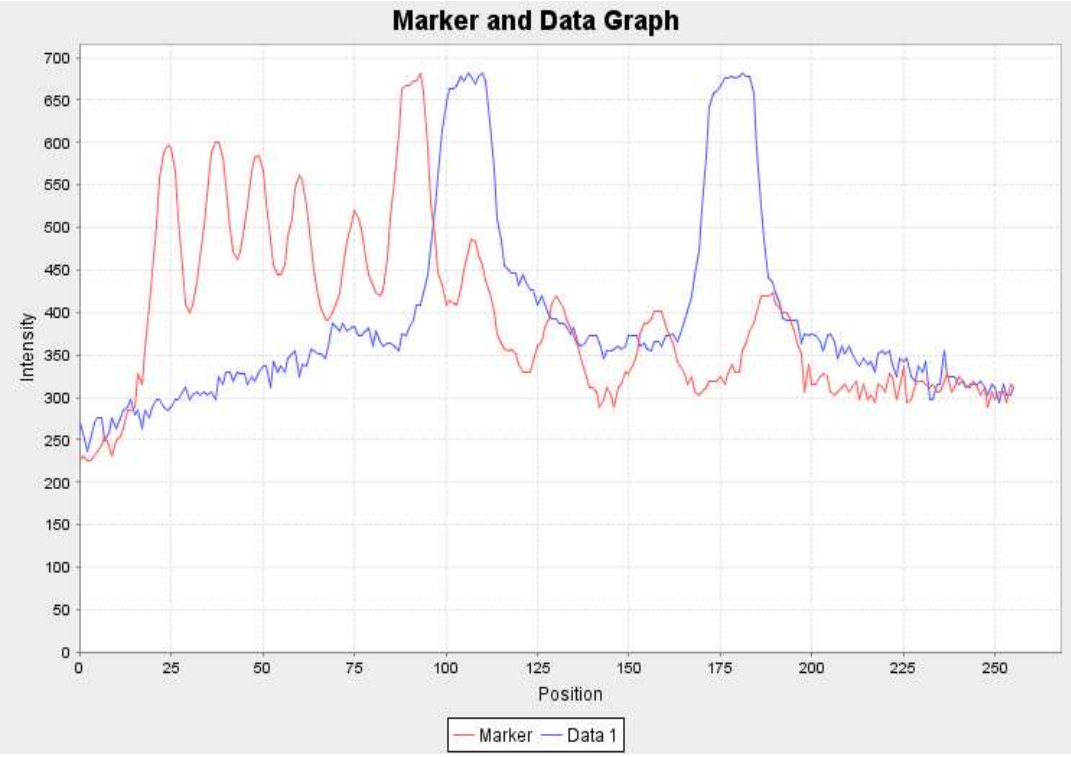
V programu je možné provádět výpočty podle čtyr různých algoritmů. Jejich výběrem je možné přizpůsobit program pro zpracování různých obrázků a dosáhnout rozdílného odečtu hodnot z nich.

Úkolem algoritmu v programu Elektroforeza je nalezení lokálních maxim, která odpovídají různým velikostem segmentů**.** Tato maxima jsou na obrázku vidět jako jednotlivé bílé pruhy a červeně jsou označena maxima odečtená programem (Obr. 4.1):



Obr. 4.1

Na obrázku 4.2 jsou zobrazeny vstupní hodnoty, se kterými algoritmus počítá. Jedná se o dvě řady hodnot (jedna jako přímka pro marker a druhá pro data), která jsou odečtena programem a vykreslena jako graf pro lepší názornost.



Obr. 4.2

Hodnota *Position* odpovídá Y – nové souřadnici v obrázku a také indexu pole, ve kterém jsou konkrétní hodnoty uloženy. Hodnota *Intensity* je získána programem. Odpovídá barevné informaci jednotlivých pixelů. Toto číslo je složeno z intensity všech tří základních barev (R (red) – červená, G (green) – zelená, B (blue) - modrá). Každé číslo nabývá hodnoty od 0 do 255. Přestože jsou obrázky černobílé, jejich odstín je složen právě z těchto tří barev.

Vstupem do všech algoritmů jsou, kromě pole dat, také proměnné *Sensitivity* (citlivost) a *Step* (krok). Obě tyto proměnné může uživatel před samotným výpočtem upravit.

Sensitivity udává citlivost, se kterou program počítá. Odpovídá jakémusi prahovému rozdílu hodnot, které algoritmus porovnává. Pro obrázky s výraznými rozdíly (jako například Obr. 4.1) stačí citlivost nižší, ale pro méně výrazné (například kratší doba exposice obrázku) je třeba volit vyšší.

Step odpovídá *kroku algoritmu* ve kterém hodnoty porovnává. Zde volba záleží na konkrétním zpracovávaném obrázku. A to hlavně na jeho rozlišení, rozložení jednotlivých maxim (a jejich případném splývání) a výskytu šumu v obrázku.

Nyní následuje popis jednotlivých algoritmů.

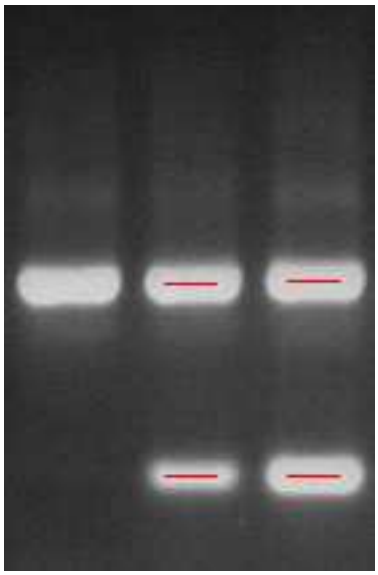
První algoritmus se nazývá *Middle*.

x = (x1 + x2) / 2
f(x1) >= (f(x+s) + i)
f(x2) >= (f(x-s) + i)

i = (hi – low) / intensity

hodnota s odpovídá proměné *step*
hodnota *i* odpovídá výsledku rozdílu nejintenzivnějšího bodu křivky (*hi*) a nejméně intenzivního (l*ow*) děleného proměnou *intensity*

Jeho úkolem je hledání určitému prahového rozdílu na vzestupu a sestupu křivky, odečtení polohy těchto hodnot a následný výpočet středu těchto hodnot (Obr. 4.3):



Obr. 4.3

Tento algoritmus je hlavně užitečný pro případy, kdy jsou jednotlivé bandy hodně široké a je třeba aproximace k nalezení jejich středu.

Algoritmus prochází požadované pole a v cyklu porovnává dvě hodnoty od sebe vzdálené o daný krok (*Step*). Pokud se jejich rozdíl intenzity rovná, nebo je větší než daná hodnota (vypočtená z rozptylu minima a maxima v poli a proměnné *Intensity*), je toto číslo uloženo. Potom se hledá druhé (první pro vzestup a druhé pro sestup křivky). Pro tuto dvojici jsou zjištěny a zapsány jejich indexy v poli. Následně je spočten jejich střed a ten je uložen mezi výsledky. Poté algoritmus pokračuje v cyklu až do konce daných dat.

Nalezené výsledky (stejně jako u následujících algoritmů) jsou vykresleny do obrázku tenkou červenou přímkou a také zapsány do tabulky výsledků (*Output Marker* případně *Output Band*).

Druhý algoritmus, *Descending*, také pracuje s proměnnými *Intensity* a *Step*. Jejich význam je stejný jako v předchozím případě.

$f(x) > (f(x+s) + i)$

Jeho princip spočívá ve hledání hodnot, pro něž platí, že následující prvek (vzdálený o *Step*) je menší o danou hodnotu (určenou výpočtem pomocí intenzity). Hledají se tedy hodnoty, po kterých následuje pokles. Tyto hodnoty jsou uloženy a je z nich vypočtena největší, která odpovídá lokálnímu maximu.

Výsledkem je tedy pole dat obsahující skutečná maxima. Tato metoda je vhodná pro rozmazanější obrázky, případně úzká maxima blízko u sebe.

Třetí algoritmus, *Ascending,* je velmi podobný druhému.

$f(x) > (f(x - s) + i)$

Pouze oproti němu hledá hodnoty odpovídající rozdílu od vzestupu. Hledá tedy hodnoty před kterými byl odpovídající nárůst intenzity.

Čtvrtý algoritmus je převzatý z literatury a implementovaný do programu. Jedná se o robustní hledání maxim (Rmax).

f(x) je lokální maximum, pokud pro všechna *k* z intervalu *x-s* do *x+s* plati:

f(x) => f(k) a dále f(x) > i + min f(k)

Uživatel může provést rozdílné nastavení parametrů algoritmu pro výpočet datových linií a pro linii markeru. Toto je velmi užitečné pokud značky markeru a bandů mají velmi rozdílnou intenzitu.

5 Program Elektroforeza

Tato kapitola pojednává o samotném programu a jeho ovládání. Program je vytvořený v jazyce Java v programu NetBeans 6.1. Pro jeho spuštění je potřeba, aby měl uživatel nainstalovaný balík *Java SE Development Kit* (JDK) nebo *Java Runtime Environment* (JRE) od společnosti Sun Microsystems. Balík je volně stažitelný na stránkách této společnosti.

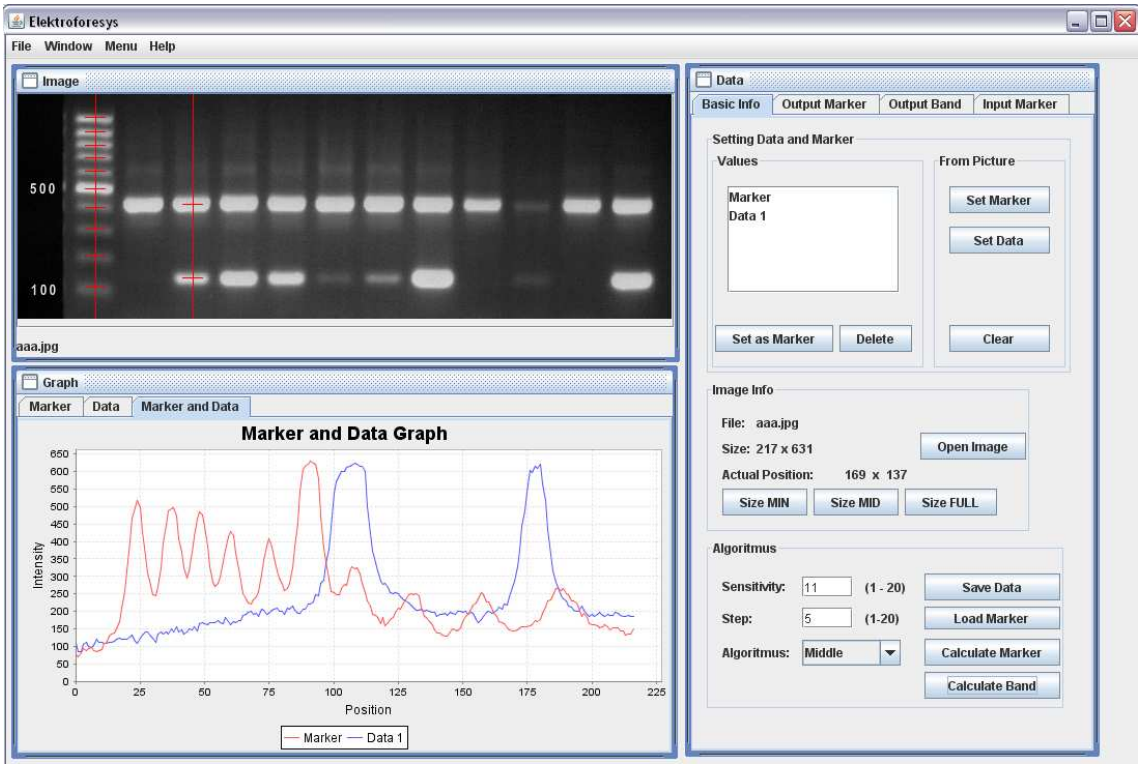
Například: http://java.sun.com/javase/downloads/index.jsp

Pro spuštění z příkazového řádku (v případě problémů) :

java -jar "C:\ElektroforezaWin.jar"

5.1 Grafické prostředí

Po spuštění uživatele přivítá úvodní obrazovka (Obr. 5.1). Ta obsahuje 3 důležitá okna**:** Image, Data, Graph (budou podrobněji popsána dále) a jednoduchý menu panel.

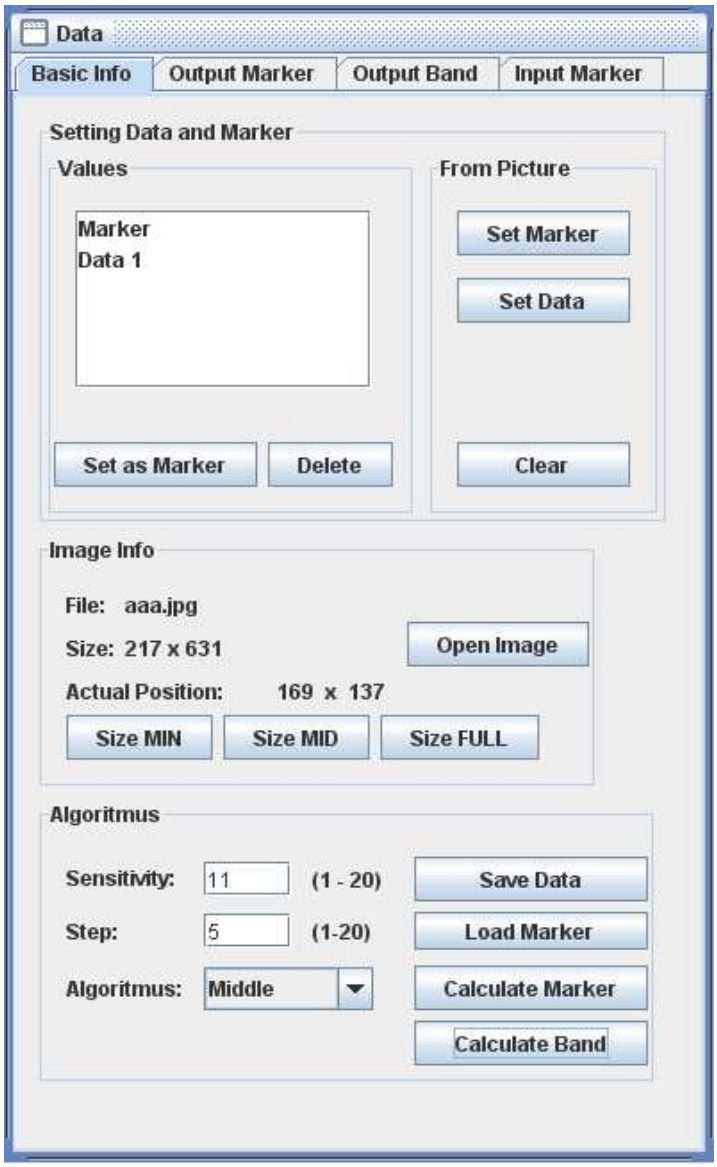


Obr. 5.1

V těchto 3 komponentách (a jejich jednotlivých záložkách) se nacházejí veškerá potřebná nastavení a vstupní i výstupní údaje, které uživatel potřebuje k ovládání programu.

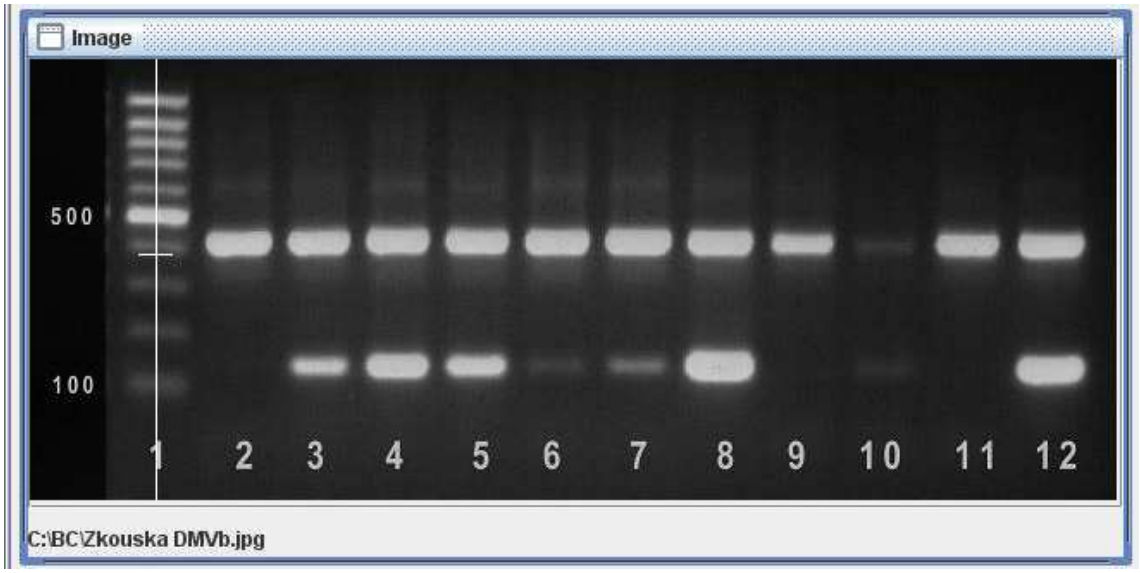
5.2 Ovládání programu

Po spuštění programu je třeba nejprve nastavit vstupní údaje. Zde je nejdůležitější panel Data (Obr. 5.2):



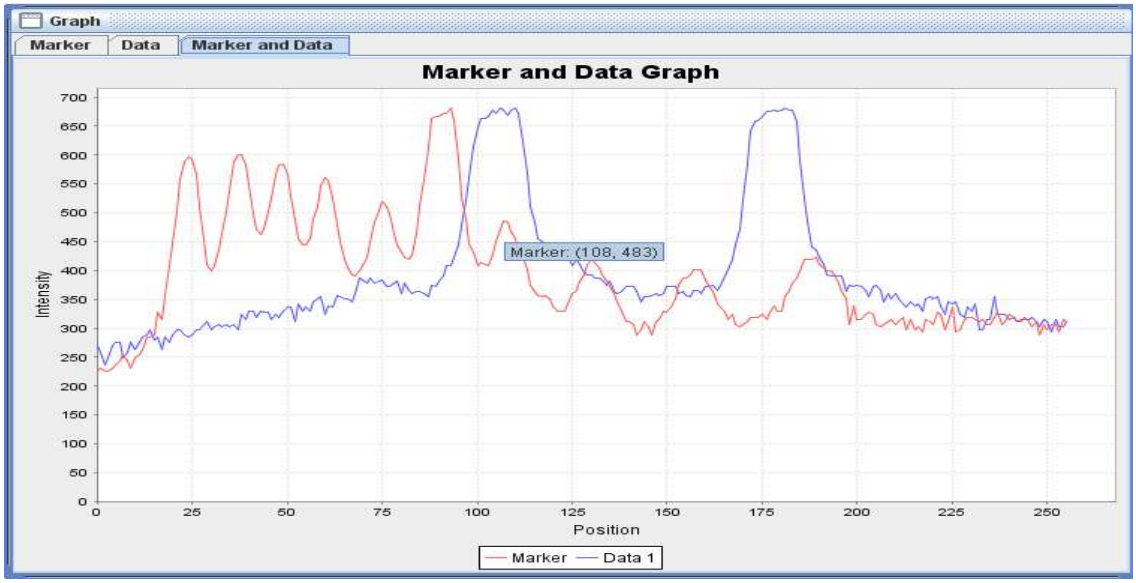
Obr. 5.2

Na něm je potřeba pomocí tlačítka *Open Image* vybrat požadovaný obraz pro zpracování (dostupné i z menu baru). Program je uzpůsoben pro zpracování obrazů ve formátu JPG (JPEG). Pomocí *open dialogu* vybereme požadovaný soubor a obraz se nám zobrazí v panelu *Image* (Obr. 5.3):



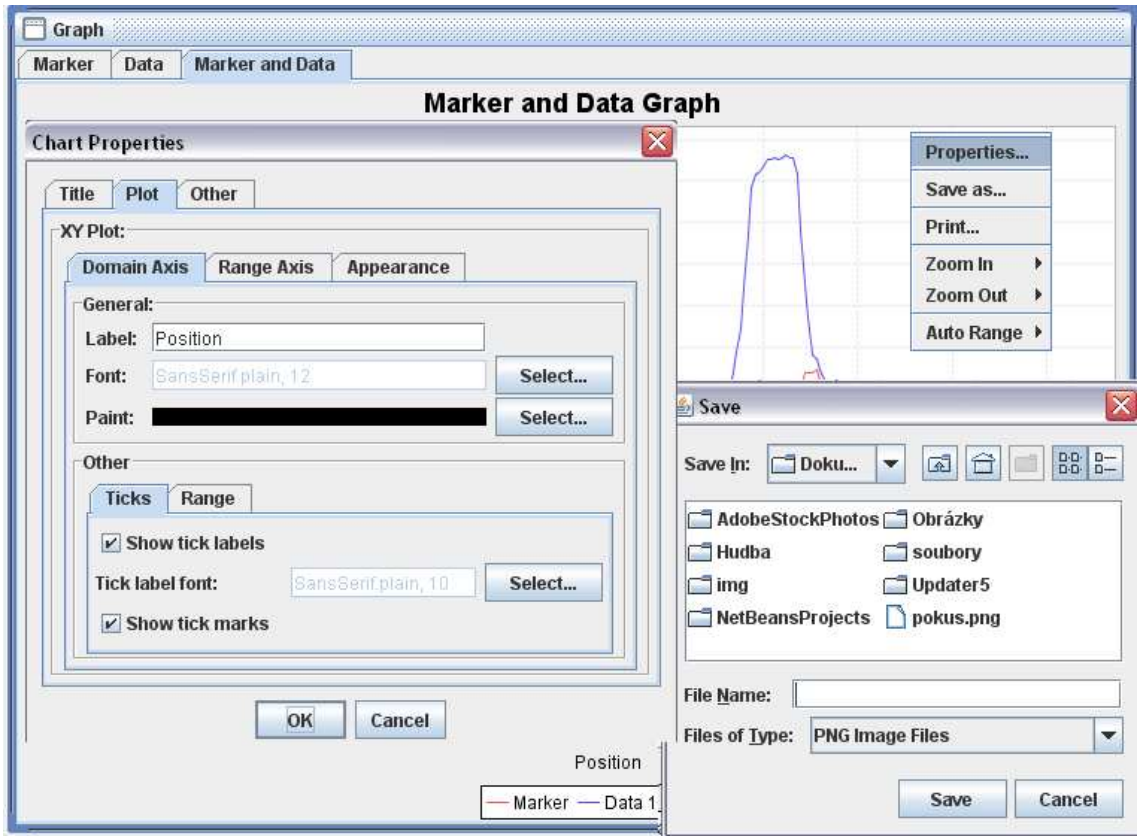
Obr. 5.3

Dalším krokem je určit programu jaká data z obrazu má zpracovávat. Toho se docílí jednoduchým kliknutím do obrazu (zobrazí se pomocná bílá přímka) a potvrzením tlačítkem *Set Marker* respektive *Set Data* (zvlášť pro oblast markeru a jednotlivé sloupečky dat). Získaná data se nám zobrazí v panelu Graph (Obr. 5.4):



Obr. 5.4

Zde jsou vykresleny křivky pro odečtené hodnoty z obrázku a uživatel má možnost prvního porovnání dat. Okno Graph nabízí další možnosti jako je *zoom* (levé tlačítko myši) a další nastavení (Obr. 5.5):

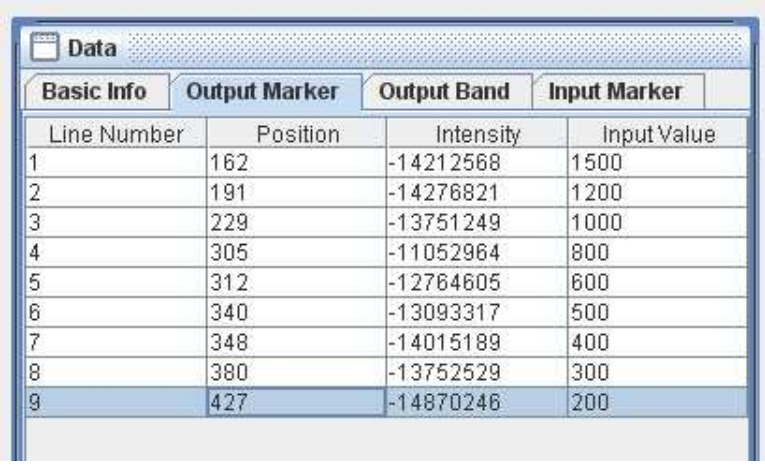


Obr. 5.5

Nyní jsou načtena námi zadaná data uvnitř programu, a ten s nimi již může provádět požadované výpočty. Program po spuštění automaticky načte defaultní nastavení (ukázkový obrázek a data pro marker). Pokud je potřeba, můžeme nahrát jiný soubor definující potřebný marker. To lze provést pomocí tlačítka *Load Marker*. Program načítá soubory s koncovkou *mar*. Potřebný soubor se dá jednoduše vytvořit z textového dokumentu změněním koncovky. Obsah souboru jsou jednotlivé hodnoty, seřazené od největší po nejmenší, odděleny středníkem (500;400;300...). Načtená data se zobrazí v záložce *Input Marker Data*.

Po nastavení všech vstupních dat můžeme přesunout pozornost k samotnému výpočtu. Ten provedeme pomocí tlačítek *Calculate Marker* a *Calculate Band*. Vypočtené hodnoty se nám zobrazí v záložkách *Output Data Marker* a *Output Data Band*. Dále máme k dispozici pole *Sensitivity* a *Step*, která nám umožňují úpravu algoritmu, který počítá požadované hodnoty. Funkce těchto konstant bude blíže specifikována při popisu algoritmu.

Je zde také možnost ručního zadání pozice markeru. Jednoduchým kliknutím do obrázku na požadovaný band markeru a následným kliknutím do tabulky v odstavci *Position* zapíšeme změněnou hodnotu (Obr. 5.6). Pak stačí už jen přepočítat hodnoty pro neznámé bandy kliknutím na *Calculate Band*.



Line Number	Position	Intensity	Input Value
1	162	-14212568	1500
2	191	-14276821	1200
3	229	-13751249	1000
4	305	-11052964	800
5	312	-12764605	600
6	340	-13093317	500
7	348	-14015189	400
8	380	-13752529	300
9	427	-14870246	200

Obr. 5.6

5.3 Implementace

5.3.1 elImage

Tato třída se stará o práci s obrázkem a jeho vykreslení v programu. Jako první bych popsal metodu sloužící k otevření a zobrazení obrázku (Obr. 5.7):

```
public void loadPicture(String file, Data dtData) {
    jLabel2.setText(file);                  //set file path in image window
    this.data = dtData;

    try {

        File file1 = new File(file);
        originalImg = ImageIO.read(file1);
        image = ImageIO.read(file1);
        imageOld = ImageIO.read(file1);
        jLabel1.setIcon(new ImageIcon(image));
    } catch (IOException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error during loading file",
                "Image frame", JOptionPane.ERROR_MESSAGE);
    }

    int h = image.getHeight();
    int w = image.getWidth();
    data.setSizePictureInfo(h, w);
    data.setFilePictureInfo(file);
```

Obr. 5.7

V této metodě je načten obrázek z parametrů získaných z *Open Dialogu* vyvolaného tlačítkem *Open Image*. Dále je načtená cesta zobrazena pod obrázkem (jako informace pro uživatele, s kterým konkrétním obrázkem se zrovna pracuje). Metoda dále zjistí výšku (*get.Height*) a šířku (*get.Width*) obrázku. Tyto údaje jsou důležité pro další práci programu.

Následuje popis metody *paint_output*, která slouží k vykreslení spočtených dat do obrázku (Obr. 5.8):

```
public void paint_output(int[] arrPositions, int intB) {
    for (int i = 0; i < arrPositions.length; i++) {
        for (int j = -10; j < 10; j++) {
            if (arrPositions[i] > 15 && arrPositions[i] < image.getWidth() - 15) {
                image.setRGB(intB + j,arrPositions[i],  Color.RED.getRGB());
```

Obr. 5.8

Tato metoda vykreslí tenkou přímku na pozici, kterou jí dodá algoritmus. Tato přímka je vykreslena červenou barvou (*Color.RED*) pro lepší viditelnost. Je zde též vidět ošetření případu, kdy se pozice nalézá u okrajů obrázku.

5.3.2 Algoritmus

Třída Algoritmus se stará o většinu výpočtů, které se s daty provádí. Je propojena s třídou Data, pomocí níž se provádí nastavení vstupních a výstupních parametrů.

```
    public int[] alg3(int sensitivity, int step,int [] poleInt) {
//return array
        int[] arrReturnMax = new int[100];
        //step array
        int[] arrTempStep = new int[2 * step];

        //setting
        int middle = 0;
        int lowest = Integer.MAX_VALUE;
        int highest = -Integer.MAX_VALUE;
        for (int d : poleInt) {
            lowest = Math.min(lowest, d);
            highest = Math.max(highest, d);
            middle = (lowest - highest) / -sensitivity;
        }

        //help variables
        int j = 0;
        int l = 0;

         //main loop
         for (int i = 0; i < (poleInt.length - step); i++) {
             //if temp array is empty
             if (arrTempStep[0] == 0) {
                 //ascending place
                 if (((poleInt[i] - poleInt[(i + step)]) <= (-middle))) {
                     arrTempStep[l] = i;
                     l++;
                 }
             } else {
                 //descending place
                 if (((poleInt[(i + step)] - poleInt[i]) <= (-middle)) &&
                         ((poleInt[(i + step + 1)] - poleInt[i + 1]) > (-middle))) {
                     arrTempStep[l] = i + step;
                     l++;
                     arrReturnMax[j] = Math.round(((arrTempStep[1] + arrTempStep[0]) / 2));
                     j++;
                     // fill arrTempStep with 0
                     for (int v = 0; v < arrTempStep.length; v++) {
                         arrTempStep[v] = 0;
                         l = 0;
                     }
                 }
             }
         }

        return arrReturnMax;

    }
```

Obr. 5.9

Na obrázku 5.9 je vidět jeden z algoritmů pro výpočet maxim křivky. Jedná se o algoritmus *Middle*. Na začátku jsou vytvořena pole potřebná pro data a vypočteno minimum (*lowest*) a maximum (*highest*) z pole vstupních dat. Dále pak přepočtena hodnota *Sensitivity* pro vstup do výpočtu. Následuje hlavní smyčka výpočtu nad polem vstupních dat. V prvním kroku je nalezena hodnota odpovídající vzestupu a uložena do pole (*arrTempStep*). V druhém kroku pak hodnota odpovídající požadovanému sestupu a také uložena. Po té je z těchto dvou hodnot vypočten střed, který je zaokrouhlen, protože musí odpovídat celému číslu, které popisuje index pole výsledků. Po tomto výpočtu je pole nulováno. Metoda vrací pole výsledků (*arrReturnMax*).

5.3.3 Data

Třída Data je nejrozsáhlejší třídou programu. Stará se o mnoho vedlejších výpočtů, převodem různých hodnot, načítáním vstupů a výstupů. Na následujícím obrázku (Obr. 5.10) je ukázána metoda, určená pro výpis hodnot nalezených programem do souboru.

```
public void SaveData() {
    try {
        //choosing file
        jFileChooser1.showSaveDialog(this);
        if(jFileChooser1.getSelectedFile() != null){
        String pathData = jFileChooser1.getSelectedFile().toString();
        PrintWriter flwriter = new PrintWriter(pathData);

        //writing to file
        for (int i = 0; i < arrValues.size(); i++) {
            if(((Values) arrValues.get(i)).getAlgPosData() !=null){
            flwriter.write(((Values) arrValues.get(i)).getName()+'\n');
            flwriter.write("Position:");
            for (int j = 0; j < ((Values) arrValues.get(i)).getAlgPosData().length; j++) {
                if(((Values) arrValues.get(i)).getAlgPosData()[j] != 0){
                flwriter.write(String.valueOf(((Values) arrValues.get(i)).getAlgPosData()[j]));
                flwriter.write(';');}
            }
            flwriter.write('\n' + "Molecular values:");
            for (int j = 0; j < ((Values) arrValues.get(i)).getAlgMolData().length; j++) {
                if(((Values) arrValues.get(i)).getAlgPosData()[j]!=0){
                flwriter.write(String.valueOf(((Values) arrValues.get(i)).getAlgMolData()[j]));
                flwriter.write(';');}
            }
            flwriter.write('\n');
            }
        }

        //closing
        flwriter.close();
        }
    } catch (FileNotFoundException ex) {
        JOptionPane.showMessageDialog(this, "Error during opening file.", "Saving data",
                JOptionPane.ERROR_MESSAGE);
    } catch (IOException ex) {
        JOptionPane.showMessageDialog(this, "Error during writing file.", "Saving data",
                JOptionPane.ERROR_MESSAGE);
    }
}
```

Obr. 5.10

Po vyvolání se zobrazí *jFileChooser* (zde typu *SaveDialog*), který ulehčuje práci se soubory. Dále jsou v různých cyklech zapisovány požadované hodnoty. Ty jsou pro lepší přehlednost odděleny řádkou ('\n') a mezi sebou středníkem. Tento formát ulehčuje zkopírování a zpracování dat například v programu Excel. Stačí jen nastavit znak pro oddělení sloupců

5.3.4 Graf

Tato třída umožňuje vykreslení dat do grafu. V programu jsou použity volně stažitelné knihovny pro vykreslování a práci s grafy (*JFreeChart Class Library*). Pro ukázku je zde část kódu pro vykreslení listu grafu s markerem (Obr. 5.11):

```
public void paintMarker(ArrayList arrMarker) {
    try {
        Values tempValues = null;
        for (Object temp : arrMarker) {
            if (((Values) temp).getType()) {
                tempValues = (Values) temp;
            }
        }
        if(tempValues != null){
        XYSeries serie = new XYSeries("MARKER");
        DefaultXYDataset defaultCategory = new DefaultXYDataset();
        for (int i = 0; i < tempValues.getData().length; i++) {
            serie.add(i, tempValues.getData()[i]);
        }
        defaultCategory.addSeries("MARKER", serie.toArray());
        JFreeChart jf = ChartFactory.createXYLineChart("Marker Graph", "Position",
                "Intensity", defaultCategory, PlotOrientation.VERTICAL, true, true, true);
        chpanelMarker = new ChartPanel(jf);
        jTabbedPane1.remove(0);
        jTabbedPane1.add(chpanelMarker, "Marker", 0);
        jTabbedPane1.setSelectedIndex(0);
        }
    } catch (Exception e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this, "Problem in painting Marker data.",
                "Error in Graph", JOptionPane.ERROR);
    }
}
```

Obr. 5.11

Kód ukazuje vytvoření nového listu typu *XY Line Chart* pomocí přiložených knihoven a nastavení jeho popisků os, názvu listu a samozřejmě naplnění daty získaných programem.

Graf umožňuje zoom pomocí levého tlačítka myši. Přes pravé tlačítko má uživatel přístup k dalším možnostem grafu jako je export obrázku ve formátu PNG, různá nastavení grafu, jeho tisk a další.

5.4 Kompenzace geometrického zkreslení

Součástí zadání bakalářské práce je i požadavek, aby program umožňoval kompenzaci geometrického zkreslení vzniklého nehomogenitou proudu. Toto není v programu implementováno, poněvadž bližší studium problému (převážně konzultace problému s pracovníky Ústavu živočišné fyziologie a genetiky) ukázalo, že přítomnost tohoto zkreslení je spíše teoretická a v praxi se téměř nevyskytuje.

Obraz elektroforézy může být zkreslen různými vlivy jako je například nestejná teplota gelu (v případě příliš vysokého napětí), jeho nehomogennost nebo přítomnost nečistot. V takových případech ale ztrácí daný pokus vypovídající hodnotu a tedy význam pro dalšího zpracování a vyhodnocení.

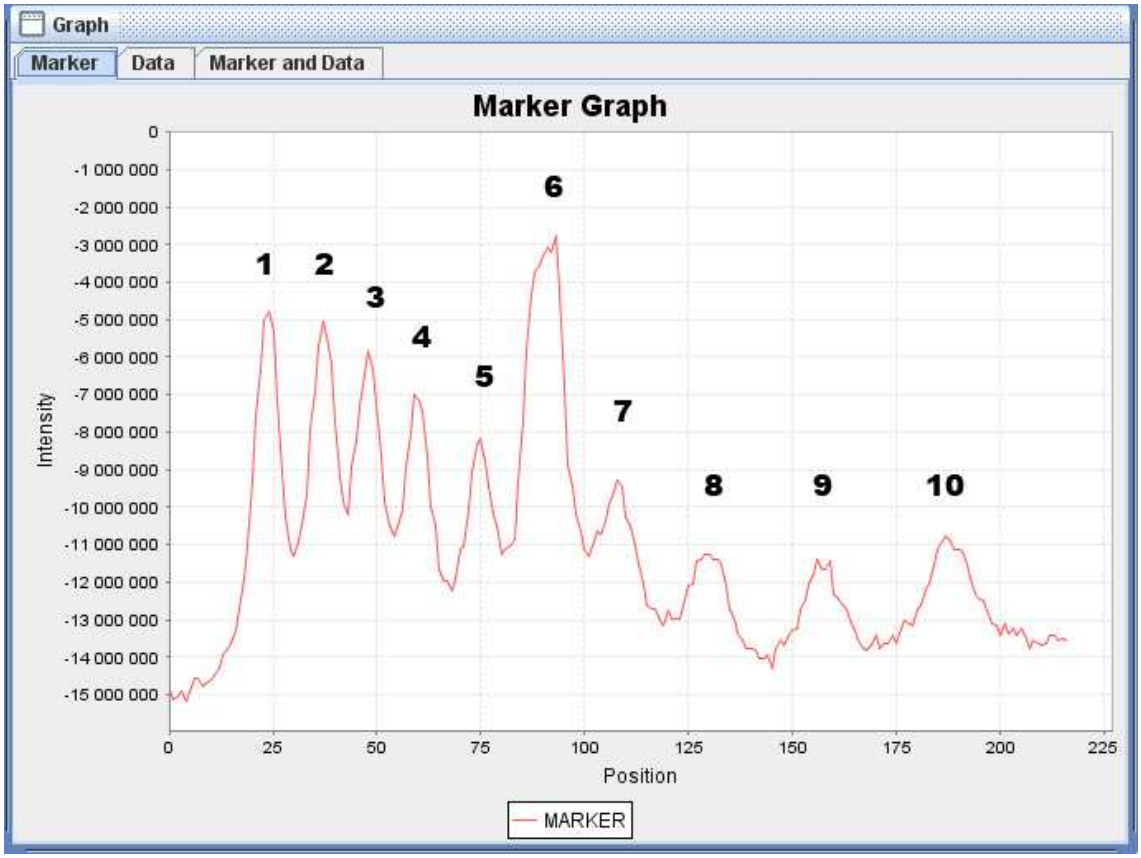
6 Experimenty

Tato kapitola popisuje různé experimenty s programem. Dále je ukázána úspěšnost jednotlivých algoritmů při prohledávání obrázku a srovnání s výsledky, které poskytuje ruční grafická metoda odečtu.

Spolu s programem je na DVD přiložen i adresář obsahující 15 obrázků z gelové elektroforézy. Na dalších řádcích bude popsána úspěšnost práce programu s těmito vzorovými obrázky.

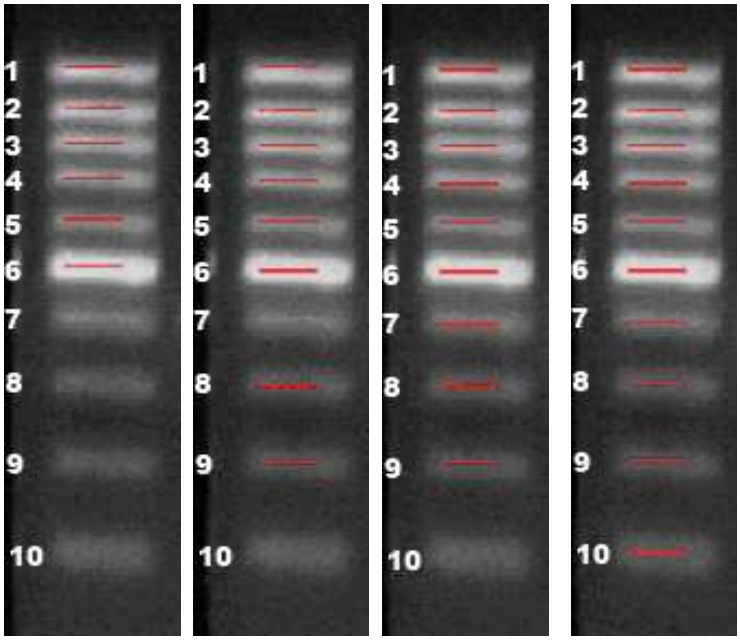
6.1 Prohledávání obrázků

Nejdříve ukážeme vliv nastavení intenzity na výsledky algoritmu. Proměnná *Intensity* odpovídá citlivosti programu.



Obr. 6.1

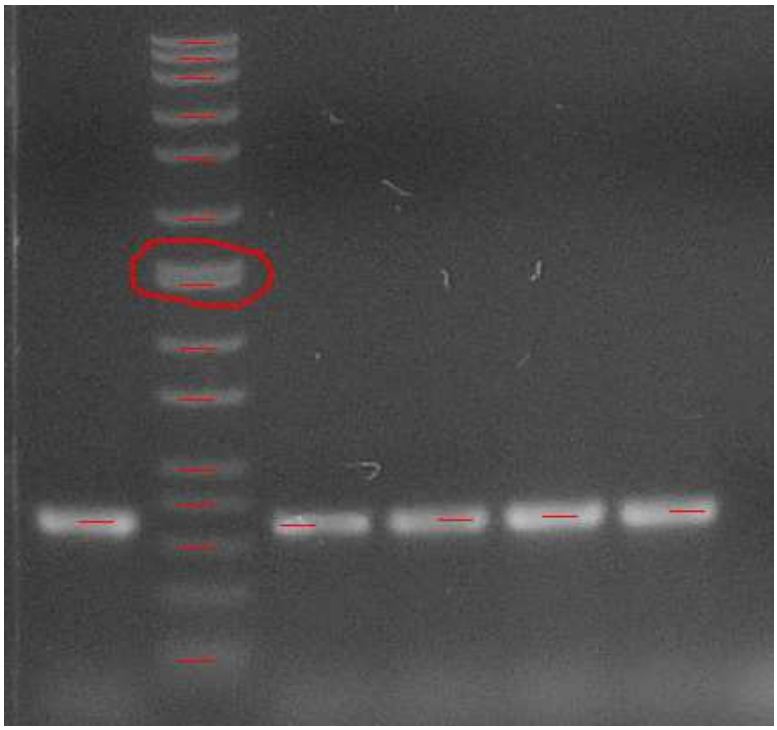
Na obrázku 6.1 jsou vidět načtená data pro marker, která jsou zobrazena v oknu *Graph*. K jednotlivým maximům jsou dopsány číslice pro snadnější orientaci.



Obr. 6.2

Na obrázku (Obr. 6.2) jsou vidět maxima odečtená algoritmem při různém nastavení senzitivity. Jednotlivé sloupce postupně odpovídají nastavení citlivosti na hodnoty 5, 8, 9 a 10. Při základním nastavení algoritmu (hodnota 5) program najde pouze prvních pět nejvýraznějších maxim. Při zvyšování senzitivity pak postupně nachází i zbývající.

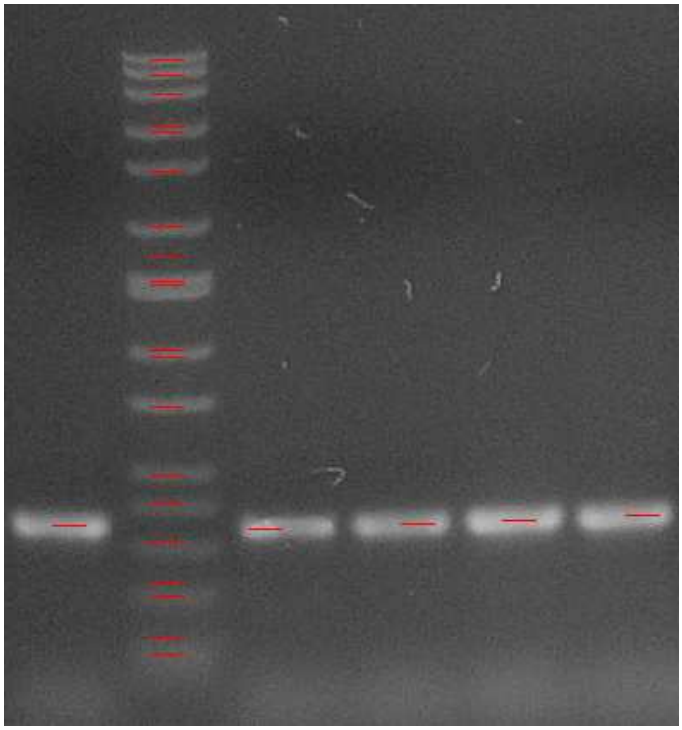
Pro všechny testované obrázky byl program schopen nalézt většinu jednotlivých bandů markeru. Problémy mohou nastat**,** když se program snaží nalézt dvě značky, které leží hodně blízko u sebe (Obr 6.3) nebo je v obrázku hodně šumu. Uživatel sice pozná, že se jedná o značky dvě, nikoli jednu, ale program není schopen tohoto dosáhnout. Jak je i na grafu pro danou linii vidět (Obr 6.4), značky barevně skoro splynou dohromady.



Obr. 6.3

Obr. 6.4

Oproti tomu pokud se budeme snažit nalézt vše a nastavíme moc velkou senzitivitu, algoritmus může najít další chybná řešení (Obr. 6.5).



Obr. 6.5

Pro práci přijde velice vhod možnost změny velikosti obrázku pomocí nastavení na MIN, MID a nebo FULL. Obzvlášť u větších obrázků se tak uživatel může rozhodnout**,** zda bude mít zobrazen jen určitý detail, nebo celý obrázek. Při změně velikosti se obrázku přepočítá rozlišení, proto je nutné zadat znovu linie pro marker a data.

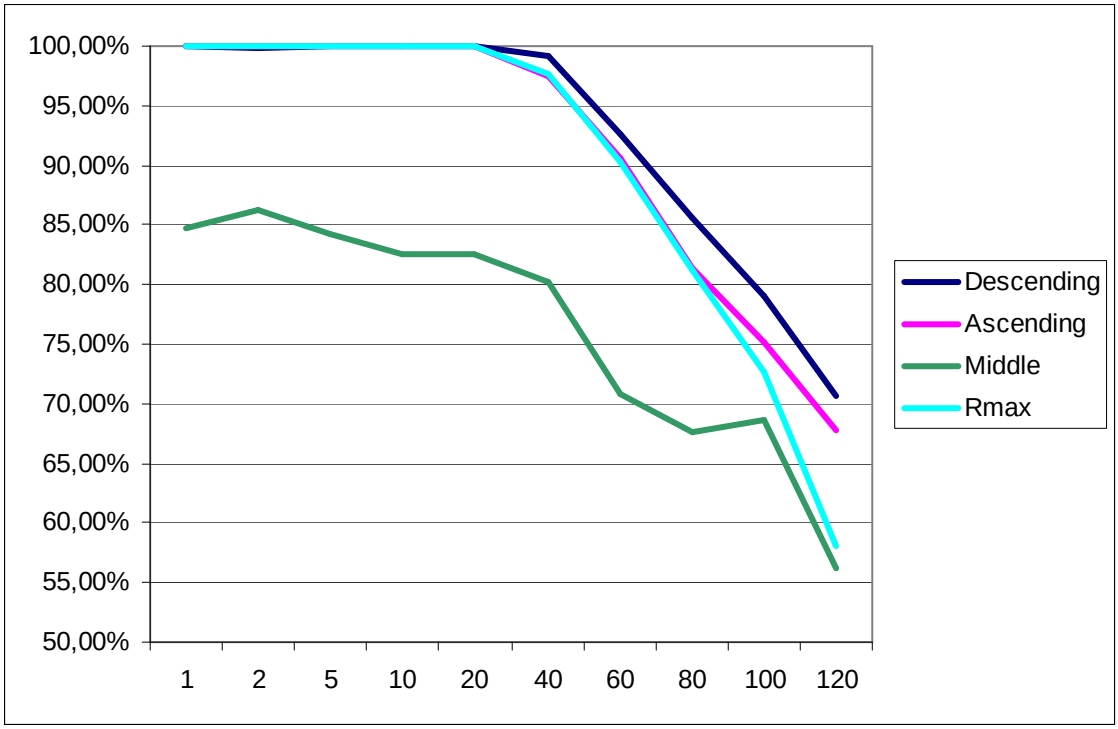
6.2 Testy algoritmů na generovaných datech

V následující kapitole je souhrn výsledků z testů jednotlivých algoritmů a na závěr statistika výsledků. Testy byly prováděny na generovaných datech. Generované křivky byly vytvořeny jako součet pěti Gaussiánů. Dále byl ke křivkám přičten šum, aby se pokus alespoň částečně přiblížil reálným vstupům. Jednotlivé Gaussiány měly pevně nastavenou šířku. Jejich amplituda byla náhodně generována mezi 200 až 765 obrazovými body. Jejich posun od počátku byl také generován náhodně. Hlídala se podmínka, aby maxima měla vhodný odstup od počátku i konce a také od sebe navzájem. Takto bylo zaručeno, že nedocházelo k jejich překrytí.

Nejprve byl proveden pokus, který ukazuje úspěšnost jednotlivých algoritmů při hledání maxim v závislosti na velikosti šumu (Tab. 6.1, Obr. 6.6). Šum byl generován jako náhodné číslo od nuly do určité hodnoty. Pro získání každé hodnoty je průměrováno měření pro sto náhodně generovaných průběhů.

Šum	Ascending	Descendig	Middle	Rmax
1	100,00%	100,00%	84,80%	100,00%
2	99,80%	100,00%	86,20%	100,00%
5	100,00%	100,00%	84,20%	100,00%
10	100,00%	100,00%	82,60%	100,00%
20	100,00%	100,00%	82,60%	100,00%
40	99,20%	97,40%	80,20%	97,60%
60	92,60%	90,60%	70,80%	90,20%
80	85,60%	81,40%	67,60%	81,20%
100	79,00%	75,20%	68,60%	72,60%
120	70,60%	67,80%	56,20%	58,00%

Tab. 6.1



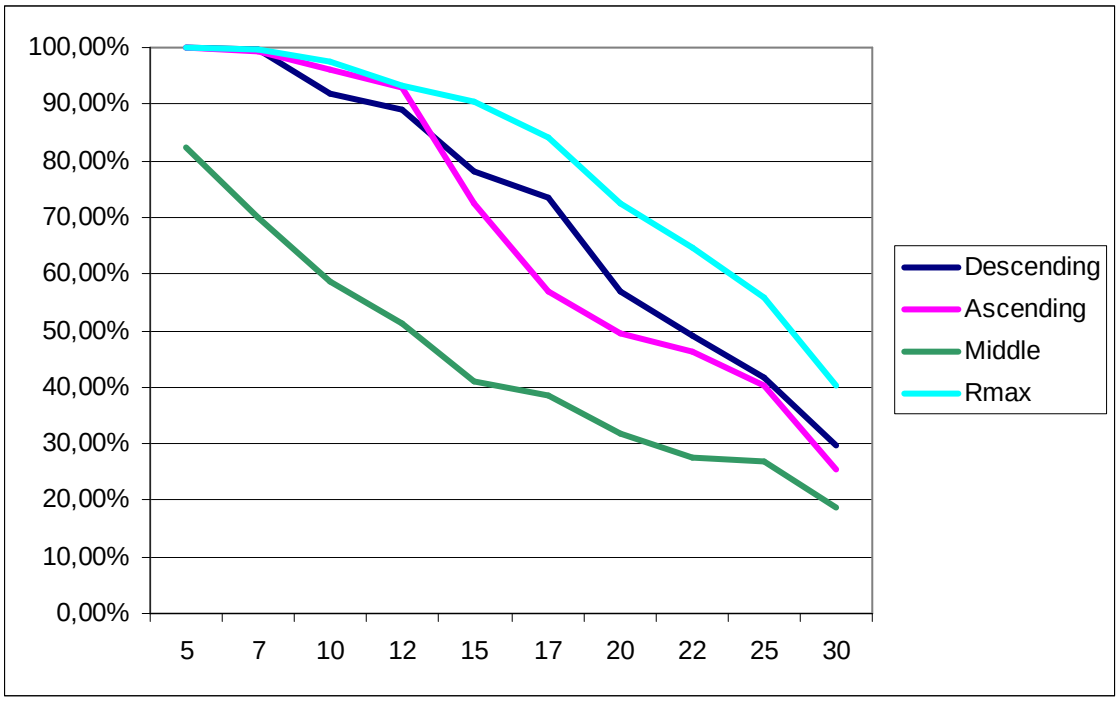
Obr. 6.6

Z výsledků je vidět, že pokud se šum drží v určitých mezích, tak algoritmy mají v podstatě stoprocentní úspěšnost. Výjimku tvoří algoritmus *Miidle*, který ovšem při přímém porovnání vstupních maxim a výsledků z algoritmů nemůže poskytovat natolik přesná data. Jinak se algoritmy *Descending* a *Ascendig* drží v úspěšnosti velmi dobře oproti referenčnímu algoritmu *Rmax*.

V dalším testu se zaměříme na závislost algoritmů nalézt maxima při změně šířky Gaussianů (Tab. 6.2 a Obr. 6.7):

Šířka maxim	Ascending	Descendig	Middle	Rmax
5	100,00%	100,00%	82,20%	100,00%
7	99,60%	99,40%	69,80%	99,60%
10	92,00%	96,00%	58,80%	97,60%
12	89,00%	93,00%	51,20%	93,20%
15	78,20%	72,40%	41,00%	90,60%
17	73,60%	56,80%	38,40%	84,20%
20	56,80%	49,60%	31,80%	72,40%
22	49,00%	46,20%	27,40%	64,60%
25	41,60%	40,40%	27,00%	55,80%
30	29,60%	25,40%	18,60%	40,40%

Tab. 6.2



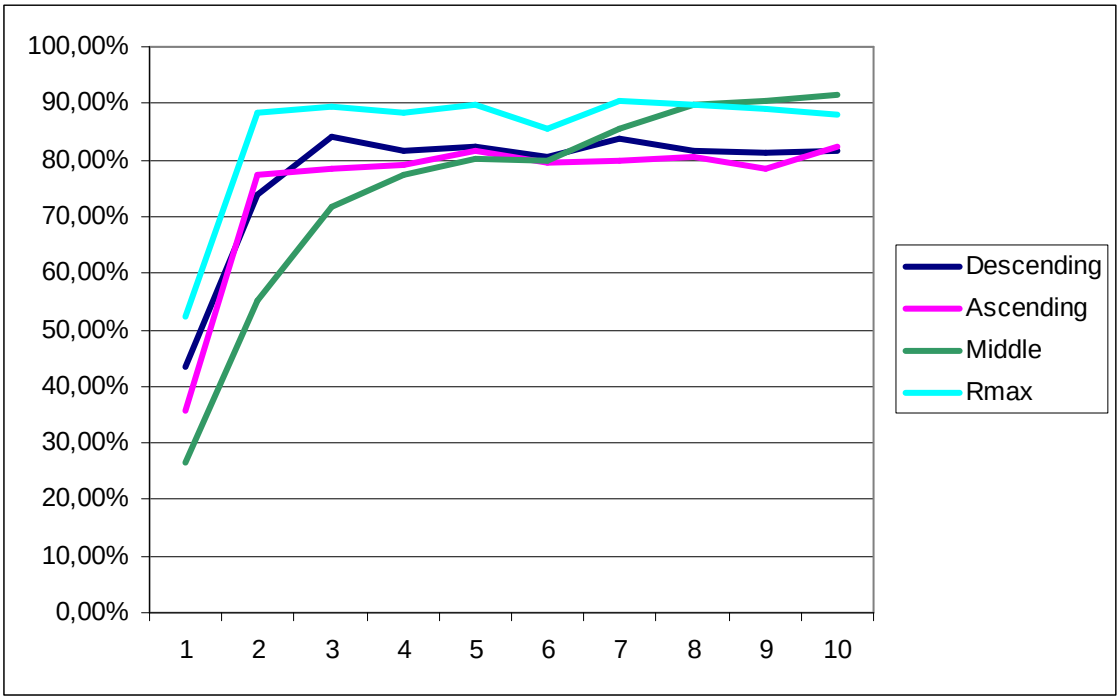
Obr. 6.7

Z těchto výsledků je názorně vidět závislost všech algoritmů na vstupních konstantách (*Sensitivity* a *Step*). Při velkém nárůstu šířky maxim úspěšnost všech algoritmů rapidně klesá.

V posledním testu jsem se zaměřil na test přesnosti nalezených dat. Při výpočtech je vstupní křivka zkreslena šumem a zvětšenou šířku Gaussianu. Postupně je zvětšována tolerance pro určení korektně nalezeného maxima. Na začátku musí být data přesná a postupně se tolerance zvětšuje i jeden pixel na obě strany od konkrétních generovaných maxim (Tab. 6.3 a Obr. 6.8):

Pixely	Ascending	Descendig	Middle	Rmax
1	43,40%	35,80%	26,40%	52,20%
2	73,80%	77,40%	55,00%	88,20%
3	84,00%	78,40%	71,80%	89,40%
4	81,60%	79,20%	77,40%	88,20%
5	82,40%	81,60%	80,20%	89,60%
6	80,60%	79,60%	79,80%	85,60%
7	83,80%	79,80%	85,60%	90,40%
8	81,80%	80,60%	89,80%	89,80%
9	81,20%	78,40%	90,40%	89,20%
10	81,60%	82,20%	91,40%	88,00%

Tab. 6.3



Obr. 6.8

Z výsledků je patrné, že i když algoritmy na počátku díky šumu poskytují malé procento úspěšných výsledků, již při malé toleranci toto procento strmě stoupá. Je tedy vidět, že algoritmy vždy nenajdou přesné maximum (které může být zkresleno šumem), ale najdou velké procento výsledků, které nejsou příliš vzdálené od skutečného maxima. Pro obrázky zarušené šumem se jako nejlepší osvědčily algoritmy Middle a Rmax.

6.3 Experiment s obrázky

Další testy byly provedeny na vybraných obrázcích, u nichž jsem testoval postupně jednotlivé algoritmy a jejich úspěšnost odečtu všech pruhů zvlášť pro bandy a zvlášť pro marker. Vždy byl každý algoritmus laděn, aby dosáhl co nejlepších možných výsledků.

Obrázky byly vybrány ze složky Images:

1.	Kravy12a13.JPG
2.	obrazek_gel.jpg
3.	p2140057.jpg
4.	p7270115.jpg
5.	pb300014.jpg

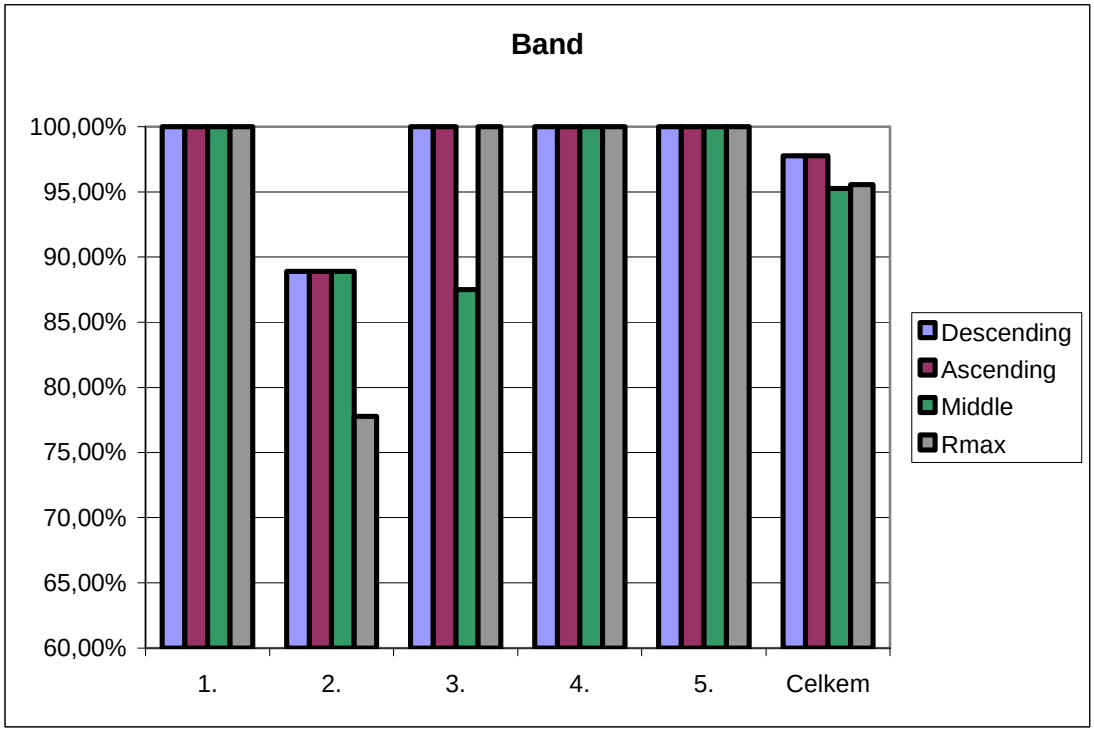
Tab. 6.4

Výsledky odečtu pomocí jednotlivých algoritmů:

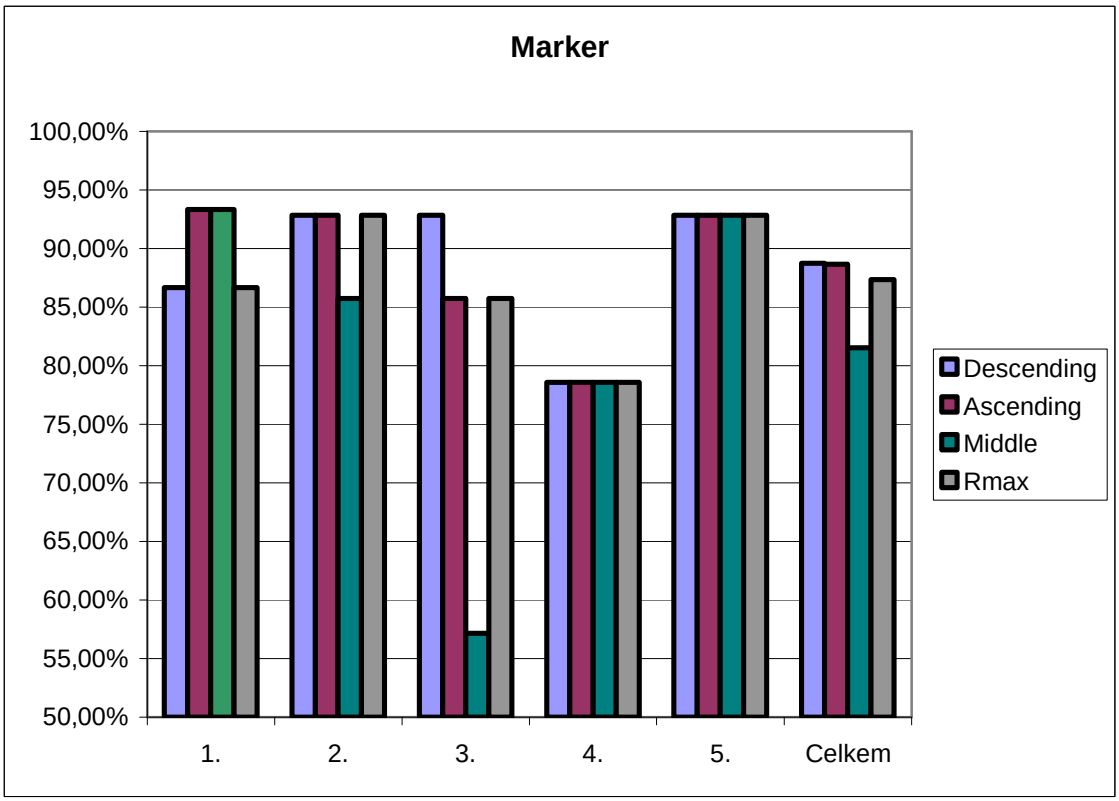
		Descending	Ascending	Middle	Rmax
1.	Band	100,00%	100,00%	100,00%	100,00%
1.	Marker	86,67%	93,33%	93,33%	86,67%
2.	Band	88,89%	88,89%	88,89%	77,78%
2.	Marker	92,86%	92,86%	85,71%	92,86%
3.	Band	100,00%	100,00%	87,50%	100,00%
3.	Marker	92,86%	85,71%	57,14%	85,71%
4.	Band	100,00%	100,00%	100,00%	100,00%
4.	Marker	78,57%	78,57%	78,57%	78,57%
5.	Band	100,00%	100,00%	100,00%	100,00%
5.	Marker	92,86%	92,86%	92,86%	92,86%
Celkem	Band	97,78%	97,78%	95,28%	95,56%
Celkem	Marker	88,76%	88,67%	81,52%	87,33%

Tab. 6.5

Na grafech Band a Marker (Obr.6.9 a Obr. 6.10) jsou pak vyneseny jednotlivé údaje. Z výsledků je vidět, že jednotlivé obrázky jsou hodně individuální a každému z nich vyhovuje jiný tip algoritmu. Na rozdíl od testů s generovanými daty je zde úspěšnost nižší, ale tyto výsledky jsme předpokládali. Obecně je úspěšnost algoritmů nad hranicí 80 % a pouze u více zašuměných nebo rozmazaných obrázků klesá.



Obr. 6.9



Obr. 6.10

6.4 Pokusná elektroforéza

Ústavem živočišné fyziologie a genetiky v.v.i., České Akademie věd v Liběchově, mi bylo umožněno provést si pokusnou gelovou elektroforézu. Tento pokus posloužil pro zkoušku programu. Elektroforéza obsahovala oproti standardnímu pokusu pět různých markerů. To umožňuje otestování různých algoritmů pro rozdílné podmínky.

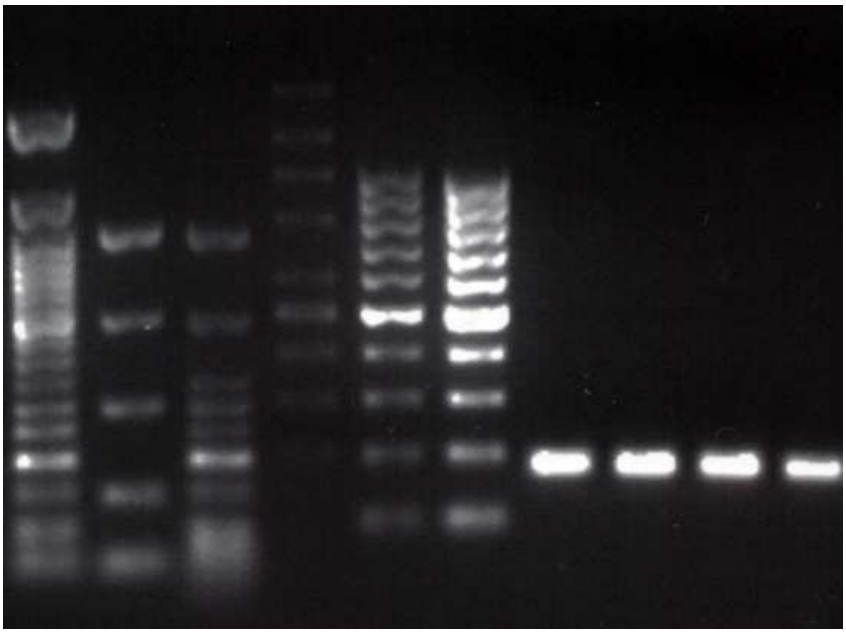
Následující tabulka (Tab. 6.6) uvádí popis proběhlé elektroforézy:

Line:	Name:
1	Biolabs 50 - BP
2	Biolabs PCR Marker
3	Biolabs Low Marker
4	200 - 1500 Marker
5	Fermentase, 100 BP, 2µl
6	Fermentase, 100 BP, 4µl
7	PCAF 1
8	PCAF 2
9	PCAF 3
10	PCAF 4

Tab. 6.6

(PCAF - gen pro histone acetyltransferase)

Výsledky pokusu (Obr. 6.11):



Obr. 6.11

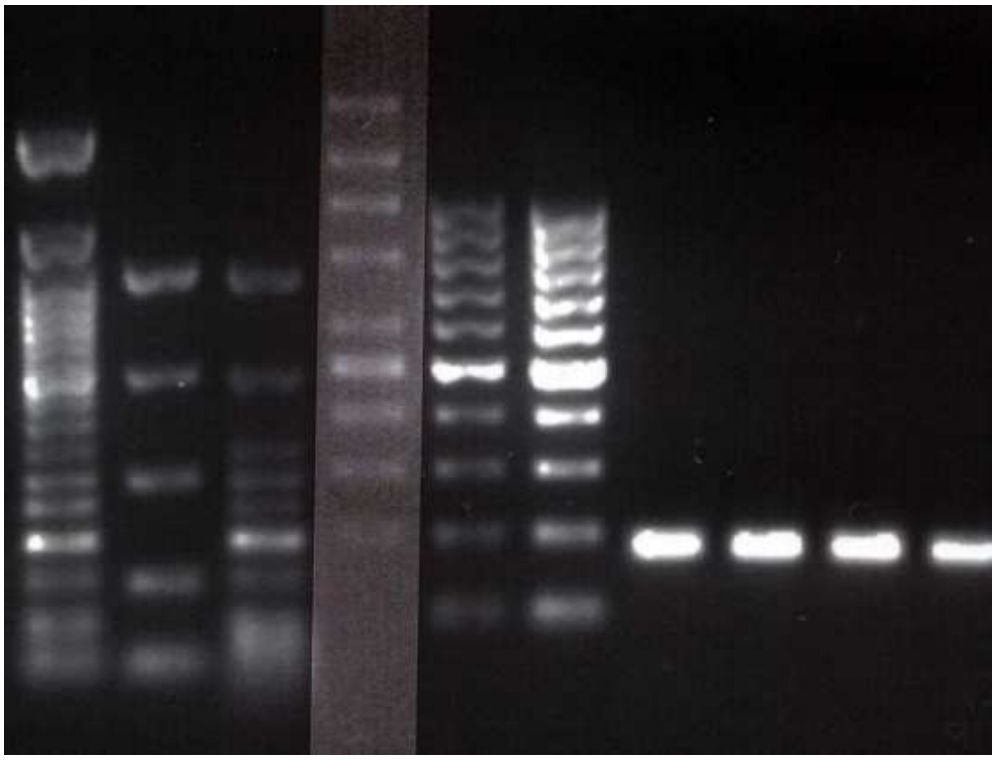
Z obrázku je patrné, že gel nebyl úplně čistý, proto jsou některé pruhy markeru zprohýbané. První marker, Biolabs 50 – BP, obsahuje celkem 17 různých bendů. Pro lepší separaci, a tedy i čitelnost, by elektroforéza musela běžet delší dobu**,** aby se jednotlivé bandy od sebe více oddělily. Čtvrtý marker je zase oproti jiným málo viditelný, bylo by třeba tedy pro další pokus použít jeho větší koncentraci.

Provedený pokus dobře demonstruje nutnost dobře volit vhodný marker pro určitý pokus a hledaný band.

Tabulka (Tab. 6.7) ukazuje výsledné hodnoty pro velikost hledaných čtyř bandů pomocí odečtu při použití různých markerů. Pro odečet markeru 200-1500 byly v obrázku, v linii tohoto markeru, provedeny úpravy pro zviditelnění dat (Obr. 6.12). Hodnoty se jinak úplně ztrácely v šumu a nebylo je možné určit.

Marker:	Data Line:	Mol. Weight:
Biolabs PCR Merker	1	200
	2	200
	3	200
	4	198
Biolabs Low Marker	1	202
	2	202
	3	202
	4	194
200 - 1500 Marker	1	202
	2	202
	3	202
	4	200
Fermentase, 100 BP, 2ml	1	196
	2	196
	3	196
	4	194
Fermentase, 100 BP,4ml	1	196
	2	196
	3	196
	4	194

Tab. 6.7



Obr. 6.12

Z tabulky je vidět, že se výsledky mírně liší, ale tyto rozdíly nejsou nikterak veliké. Hledaný pruh měl velikost 195 bp (tato velikost je známa na základě sekvenačních dat a zvolené polymerásové řetězové reakce). Podle ručního odečtu by se nacházel v rozmezí 190 – 210 v závislosti na tom, podle kterého markeru by se počítalo. Výsledky z programu jsou tedy celkem přesné, a po určité praxi v ovládání programu je jejich nalezení mnohem rychlejší, než při ručním zpracování.

7 Závěr

Program Elektroforeza je napsán s důrazem na jednoduché a přehledné ovládání. Uživatel by měl po krátké době lehce proniknout do ovládání a program bez obtíží používat.

Díky použitému jazyku Java je s malými odlišnostmi lehce přenositelný na různé platformy (odzkoušeno na Windows XP, MacOS X a Linux). Tyto odlišnosti by se daly, pokud by tomu bylo zapotřebí, lehce doladit, a program tak přizpůsobit na použití v co nejširším nasazení. Jeho náročnost na hardware počítače je dána právě javou a spuštění by nemělo být problémem na žádném běžném počítači.

Algoritmy použité pro výpočty jsou navrhovány přesně na řešení konkrétního problému. Neměly by být tedy zbytečně náročné a jejich výsledky byly odzkoušeny na dostupných datech. Program nenachází výsledky se stoprocentním úspěchem, toho by bylo dosaženo jen na konkrétních, dostatečně kontrastních obrázcích. To ostatně platí i pro komerčně dostupné programy. Důraz byl kladen na zpracování co nejširší oblasti možných případů, jelikož vstupní data jsou dosti různorodá.

Mým hlavním záměrem bylo proniknout do metodiky gelové elektroforézy a navrhnout použitelný program, který by se dal v budoucnu dále upravovat a rozšiřovat o další funkce. Nebude ani obtížné program přizpůsobit podle konkrétních požadavků případného zájemce. Program i v nynější podobě představuje možnost rychlejšího a jednoduššího zpracování výsledků gelové elektroforézy.

Práce na tomto programu mi přinesla velké zdokonalení programátorských schopností, zkušenosti s vyhledáváním a zpracováním odborné literatury, možnost návrhu algoritmů pro zpracování obrazu a v neposlední řadě přiblížení práce ve vědě. Vývoji tohoto programu bych se rád věnoval i v budoucnu. Program by bylo možné doplnit o knihovnu markerů používaných v České Republice. Také by bylo vhodné program otestovat na konkrétních fragmentech DNA známé velikosti, které by pokrývaly širokou škálu délek, vyskytujících se v praxi. Opakováním elektroforéz by bylo možné získat

data pro statistickou analýzu přesnosti programu v konkrétních laboratorních podmínkách.

8 Zdroje

8.1 Seznam použité literatury

Adrian, T. and Heinrich, W. COMAP: a comigrating analysis program for estimating the relationship of adenoviruses on the genome level. Nucleic Acids Res. 1986; 14(1):559-65.

Eckel Bruce, Myslíme v jazyku Java – Knihovna programátora. Grada, 2001

Eckel Bruce, Myslíme v jazyku Java – Knihovna zkušeného programátora. Grada, 2001

Gariepy, C. E.; Lomax, M. I., and Grossman, L. I. SPLINT: a cubic spline interpolation program for the analysis of fragment sizes in one-dimensional electrophoresis gels. Nucleic Acids Res. 1986; 14(1):575-81.

Gough, E. J. and Gough, N. M. Direct calculation of the sizes of DNA fragments separated by gel electrophoresis using programmes written for a pocket calculator. Nucleic Acids Res. 1984; 12(1-2):845-53.

Herout Pavel, JAVA – Grafické uživatelské prostředí a čeština. Kopp, 2004

Herout Pavel, Učebnice jazyka JAVA. Kopp, 2003

Maina, C. V.; Nolan, G. P., and Szalay, A. A. Molecular weight determination program. Nucleic Acids Res. 1984; 12(1-2):695-702.

Maniatis, T., Fritsch, E. F. and Sambrook, J. Molecular Cloning: A Laboratory Manual (1982)

Metzker, M. L.; Allain, K. M., and Gibbs, R. A. Accurate determination of DNA in agarose gels using the novel algorithm GelScann(1.0). Comput Appl Biosci. 1995; 11(2):187-94.

Redman, T. and Jacobs, T. Electrophoretic gel image analysis software for the molecular biology laboratory. Biotechniques. 1991; 10(6):790-4.

Russell, P. J.; Crandall, R. E., and Feinbaum, R. GELYSIS: Pascal-implemented analysis of one-dimensional electrophoresis gels. Nucleic Acids Res. 1984; 12(1-2):493-8.

Russell, P. J.; Doenias, J. M., and Russell, S. J. GELYMAC: a Macintosh

application for calculating DNA fragment size from gel electrophoresis migration data. Comput Appl Biosci. 1991; 7(2):265-6.

Southern, E. M. Measurement of DNA length by gel electrophoresis. Anal Biochem. 1979; 100(2):319-23.

Storchová Z. Molekuly na povel III. Jak to udělat, aby molekula byla dobře vididtelná. Vesmír. 1998; 77:372-374.

Tietz, D. Analysis of one-dimensional gels and two-dimensional Serwer-type gels on the basis of the extended Ogston model using personal computers. Electrophoresis. 1991; 12(1):28-39.

Yeoh, H. H. Estimation of DNA restriction fragment size using a PC. Comput Biol Med. 1991; 21(1-2):51-4.

8.2 Seznam použitého softwaru

NetBeans IDE 6.1

NetBeans IDE 6.5

Microsoft Word 2002

Microsoft Excel 2002

Adobe Photoshop

8.3 Další zdroje

http://java.sun.com/javase/technologies/desktop/media/

https://jai-core.dev.java.net/

http://www.java-tips.org/

http://forums.sun.com/

http://www.leepoint.net/notes-java/index.html

http://www.zaachi.com/cs/category/blog/java/

http://www.netbeans.org/kb/index.html

9 Přílohy

Přiložené CD obsahuje:

/Elektroforeza	- samotný program
/Elektroforeza_src	- zdrojový kód programu
/Images	- obrázky pro program
/Markers	- soubory s přednastavenými markery
/JavaDoc	- javadoc k programu
/Others	- návod, Classdiagram, zadání bakalářské práce
/Java	- pomocné soubory pro Javu