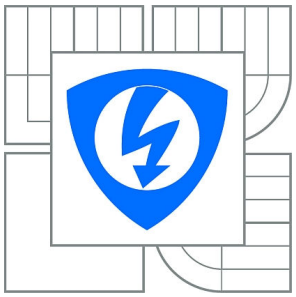
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF RADIO ELECTRONICS

FRAKTÁL V SEKVENCI DNA

FRACTAL OF DNA SEQUENCE DATA

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

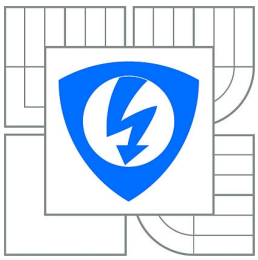
AUTOR PRÁCE — JIŘÍ NEDVĚD

AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN VALLA

SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Bakalářská práce

bakalářský studijní obor
Elektronika a sdělovací technika

Student: Jiří Nedvěd ***ID:*** 106665
Ročník: 3 ***Akademický rok:*** 2009/2010

NÁZEV TÉMATU:

Fraktál v sekvenci DNA

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte a podrobně popište využití fraktálové analýzy k vytvoření systému pro klasifikaci DNA. V programu MATLAB implementujte výpočet dimenze a multifraktálního koeficientu. Funkčnost vytvořeného programu pečlivě ověřte.

DOPORUČENÁ LITERATURA:

[1] ZU-GUO, Y., ANH, V. Fractals in DNA sequence analysis. IOP electronic journals : Chinese Physics [online]. 2002, is. 12 [cit. 2002-07-20], s. 1313-1318.

[2]GARIAEV, Peter , et al. Fractal Structure in DNA code and human language: Towards a semiotics of biogenetic information. In International Journal of Computing Anticipatory Systems. [s.l.] : [s.n.], 2002. s. 255-273. ISBN 2-9600262-7-6. ISSN 1373-5411 .

Termín zadání: 8.2.2010 ***Termín odevzdání:*** 28.5.2010

Vedoucí práce: Ing. Martin Valla

prof. Dr. Ing. Zbyněk Raida
Předseda oborové rady

Abstrakt

Práce se zabývá hledáním fraktální struktury v sekvenci DNA, která bude vykreslena ve čtverci ACGT. Na počátku jsou rozebrány fraktály obecně a důkladně je probrána konstrukce Sierpinského trojúhelníku deterministickou metodou a metodou chaos game. Následuje kapitola, kde je přiblížena struktura a sekvence DNA. Jsou zde také uvedeny již známé fraktální struktury nacházející se v DNA sekvenci, nebo se jí bezprostředně týká. Dále je uvedena úprava metody chaos game ze Sierpinského trojúhelníku na čtverec ACGT, a do něj bude vykreslena mapa dané sekvence DNA. Do čtverce ACGT se následně vykreslí vybrané sekvence DNA. Obrázky sekvencí se dále zpracují metodami BCM a PSM.

Klíčová slova:

Fraktál, soběpodobnost, sekvence DNA, konstrukce Sierpinského trojúhelníku, BCM, Box Counting Method, PSM, Power Spectrum Method

Abstract

This project is looking for fractal structury in DNA sequence data, which be maped in square ACGT. First, this project is introduced in problematics of fractals and targeted on making of Sierpinski gasket. Second, the project is introduced in DNA and DNA sequence. Next is sketched now known fractal's structures in DNA sequence data or fractal's structure closed DNA sequence. Then will changed metod chaos game of Sierpinski gasket to square ACGT, in where will drawen map of DNA sequences. Next, these pictures will processed by Box Counting Method and Power Spectrum Method.

Key words:

Fractal, self-simmilar, DNA sequence, construction of Sierpinski gasket, BCM, Box Counting Method, PSM, Power Spectrum Method

Bibliografická citace:

NEDVĚD, J. *Fraktál v sekvenci DNA.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 45 s. 7 příl. Vedoucí bakalářské práce Ing. Martin Valla.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma Fraktál v sekvenci DNA 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 práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 27. května 2010 .....
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Martinu Vallovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne 27. května 2010 .....
podpis autora

Obsah:

1. ÚVOD

Proč zrovna fraktální analýza DNA sekvencí? Fraktály mají zajímavou vlastnost, že je lze popsat jednoduchými matematickými rovnicemi, a přesto dostaneme zajímavé nebo až nepředvídatelné výsledky. Pokud je necháme vykreslit (viz. **Obrázek 1.1**) tak nejen, že zjistíme, že jsou na pohled estetické (po nadefinování barev), ale i po přiblížení můžeme najít stejný vzor, jaký jsme viděli na počátku (více v kapitole 1.1. Seznámení s fraktály a v kapitole 2. Rozdělení fraktálů). [1]

V DNA sekvenci lze najít něco podobného. Na zemi žije mnoho druhů organismů na uhlíkové struktuře a v DNA mají stejné čtyři báze jako člověk A, C, G a T, ale přesto každý druh organismu vypadá jinak. Vše právě určuje DNA (podrobněji rozebráno v kapitole 4. Seznámení s DNA). [1]

DNA bude vykreslena pomocí bodů do plochy, vznikne určitý obraz, který bude dále upravován, a na něm bude hledána fraktální souvislost (více v kap. 6. Zkoumání sekvence DNA). Realizování tohoto algoritmu bude provedeno v programu Matlab (zdrojové kódy jsou uvedeny v přílohách na konci práce a na přiloženém CD).

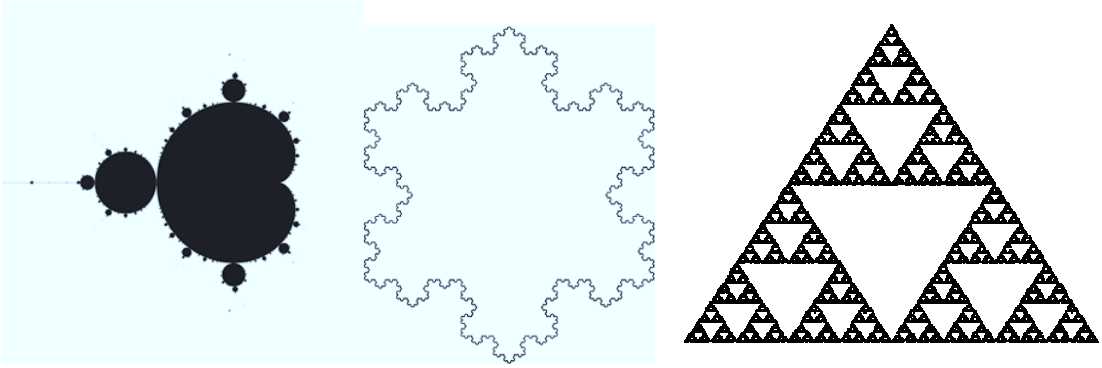
1.1. Seznámení s fraktály

Co je to fraktál? Jen málokdo si pod tím dokáže něco představit a přitom se fraktály objevují všude kolem nás, ve vesmíru, na zemi i v tělech živočichů a člověka.
Fraktál je přírodní struktura, kterou si můžeme představit jako dobře vzrostlý strom bez listí, nebo jako strukturu listu („žilek“ ze spodní strany listu), nebo řeky tekoucí kontinentem, sněhovou vločku či dokonce cévní systém v těle člověka.[2]

Fraktál je objekt, ve kterém objevíme stejnou strukturu po daném počtu přiblížení či oddálení. Tato vlastnost byla popsána jako **soběpodobnost**. Poprvé ji pozoroval francouzský matematik Benoit Mandelbrot na rušení, které se objevilo na telegrafních drátech. Také on dal této struktuře jméno **fractal** – od latinského *fractus* (rozbitý, zlomený). [2]

V dnešní době člověk spíše zná pravidelné tvary, které lze dobře geometricky popsat, ale pokud se podíváme do přírody, tak pravidelných tvarů najdeme jen velmi málo, pokud vůbec nějaký. Přesné geometrické tvary vytvořil až člověk. Příroda sama má mnoho tvarů, které lze popsat jen fraktální geometrií [3].

Fraktální geometrie se zabývá nepravidelností tvarů, obrazců a uspořádání. Mnoho fraktálů vypadá velmi efektně a v různém barevném provedení vypadají opravdu impozantně. Uvedu nyní pár známých základních a černobílých vyobrazení fraktálů.



Obrázek 1.1: Ukázky známých vyobrazení fraktálů. Vlevo Mandelbrotova množina, uprostřed Kochova sněhová vločka a vpravo Sierpinského trojúhelník. [21]

Poznámka autora: Fraktálů je samozřejmě mnoho, pokud by jste chtěli vidět různé fraktály v barevném provedení, stačí zadat do vyhledávače Google heslo fraktál, a v sekci obrázků se objeví velké množství těchto impozantních útvarů.

1.2. Dimenze fraktálů

Klasická euklidovská geometrie je definovaná na pevně daných topologických úrovních dimenzí, kdy bod má dimenzi rovnou nule (D=0), přímka jedné (D=1), plocha nebo rovina dvěma (D=2) a prostor třem (D=3). Fraktální geometrie má také definovanou dimenzi, zvanou též Hausdorffova-Besicovitchova dimenze, a ta může nabývat hodnot 0, 1, 1 až 2, 2, 2 až 3. Hodnoty dimenzí mezi 1 až 2 znamená, že fraktál vyplňuje více jednorozměrnou přímku, ale stále méně než dvojrozměrná plocha. Podobně hodnoty 2 až 3, fraktál vyplňuje více než dvojrozměrnou plochu, ale stále méně než trojrozměrný prostor. [1,2]

1.2.1. Výpočet fraktální dimenze

Fraktální dimenze D udává míru nepravidelnosti daného objektu. U klasických objektů s topologickou dimenzí vždy obvod konverguje k určité limitní hodnotě, zatímco u fraktálů se délka (nebo obvod) neustále zvětšuje. Tato vlastnost je nazývána jako Richardsonův efekt. [2]

Máme-li úsečku, tedy objekt o jedné dimenzi, a rozdělíme ji na 4 stejné dílky, lze taky říci soběpodobné dílky, pak délka dílku bude

$$r = \frac{1}{4} = 0{,}25\,. \tag{1}$$

Obecně lze tuto rovnici (1) přepsat na

$$r = \frac{1}{N}\,. \tag{2}$$

Vezmeme-li plochu, to je dvojrozměrný objekt, nebo lépe čtverec a rozdělíme jej na 16 dílků, získáme

$$r = \frac{1}{16^{\frac{1}{2}}} = \frac{1}{4} = 0{,}25\,, \tag{3}$$

což lze zase přepsat do obecné formy:

$$r = \frac{1}{N^{\frac{1}{2}}}\,. \tag{4}$$

Stejně můžeme postupovat i v prostoru pro krychli, kdy dostaneme

$$r = \frac{1}{N^{\frac{1}{3}}}\,. \tag{5}$$

Předcházející tři vzorce (2,4,5) lze spojit do jedné obecné výsledné rovnice

$$r = \frac{1}{N^{\frac{1}{D}}}\,, \tag{6}$$

kde D je dimenze objektu.

Nyní stačí vyjádřit D. To dostaneme zlogaritmováním rovnice (6)

$$\log r = -\log N^{\frac{1}{D}}, \tag{7}$$

a další úpravou rovnice (5) dostaneme výraz pro samotnou dimenzi D:

$$D = \frac{\log N}{\log\left(\frac{1}{r}\right)} \quad [-], \tag{8}$$

kde N je faktor změny délky, neboli na kolik částí se objekt bude dělit

$\frac{1}{r}$ je faktor změny měřítka (většinou je označován jako **m**)

r je velikost soběpodobné části

D je velikost dimenze.

1.2.2. Ukázka výpočtu dimenze

Ukázku výpočtu provedu podle vzorce (8) pro první dvě transformace Kochovy křivky (vyobrazeno na **Obrázek 1.2**). Při každé transformaci se délka Kochovy křivky zmenší na r=1/3 a křivku vždy rozdělíme na N=4 soběpodobné části. [3]

Obrázek 1.2: První dvě transformace Kochovy křivky. [3]

Dosazením a výpočtem vzorce (8) dostaneme hodnotu dimenze:

$$D = \frac{\log N}{\log\left(\frac{1}{r}\right)} = \frac{\log 4}{\log\left(\frac{1}{\frac{1}{3}}\right)} = \frac{\log 4}{\log 3} = 1{,}2618\,. \tag{10}$$

Z výsledku (10) lze vidět, že opravdu jde o fraktálovou strukturu, neboť dimenze vyšla D=1,26.

Pro srovnání uvedu v tabulkách pár dimenzí různých objektů.

Tabulka 1: Dimenze geometrických objektů. [4]

Objekt	**Dimenze**
Bod	0
Přímka	1
Čtverec - plocha	2
Krychle - prostor	3
Kochova křivka	1,26
Sierpinského trojúhelník	1,59

Tabulka 2: Dimenze přírodních objektů. [7]

Objekt	**Dimenze**
Povrh neerodovaných skal	2,2 – 2,3
Obvod 2D průmětu oblaku	1,33
Pobřeží	1,26

Tabulka 3: Dimenze vybraných anatomických struktur. [3]

Struktura	**Dimenze**
Průdušky	~ 3
Lidský mozek	2,73 – 2,79
Endoplazmatická šíťka	2,72
Tepny	2,7
Membrána alveoly	2,17

2. ROZDĚLENÍ FRAKTÁLŮ

- L-Systémy
- Iterační funkční systémy
- Polynomické fraktály
- Náhodné fraktály

2.1. L-Systémy

L-Systémy jsou často označovány jako fraktální křivky. Do této skupiny patří i Kochova křivka (**Obrázek 1.2**) a Kochova vločka (**Obrázek 1.1**), která vznikne ze třech Kochových křivek složených do trojúhelníku. [3, 4]

L-Systémy byly poprvé použity pro simulaci vývoje mnohobuněčných organismů. Dnes se nejvíce používají právě pro simulace v biologii, geologii a podobných přírodních vědách, kde ovšem nelze plně využít, protože přírodní struktura je složitá a dosud přesně nepopsaná. Při hlubším zkoumání se projeví nedokonalosti a ukáže se, že tato metoda není doposud plně propracovaná do detailu. [3, 4]

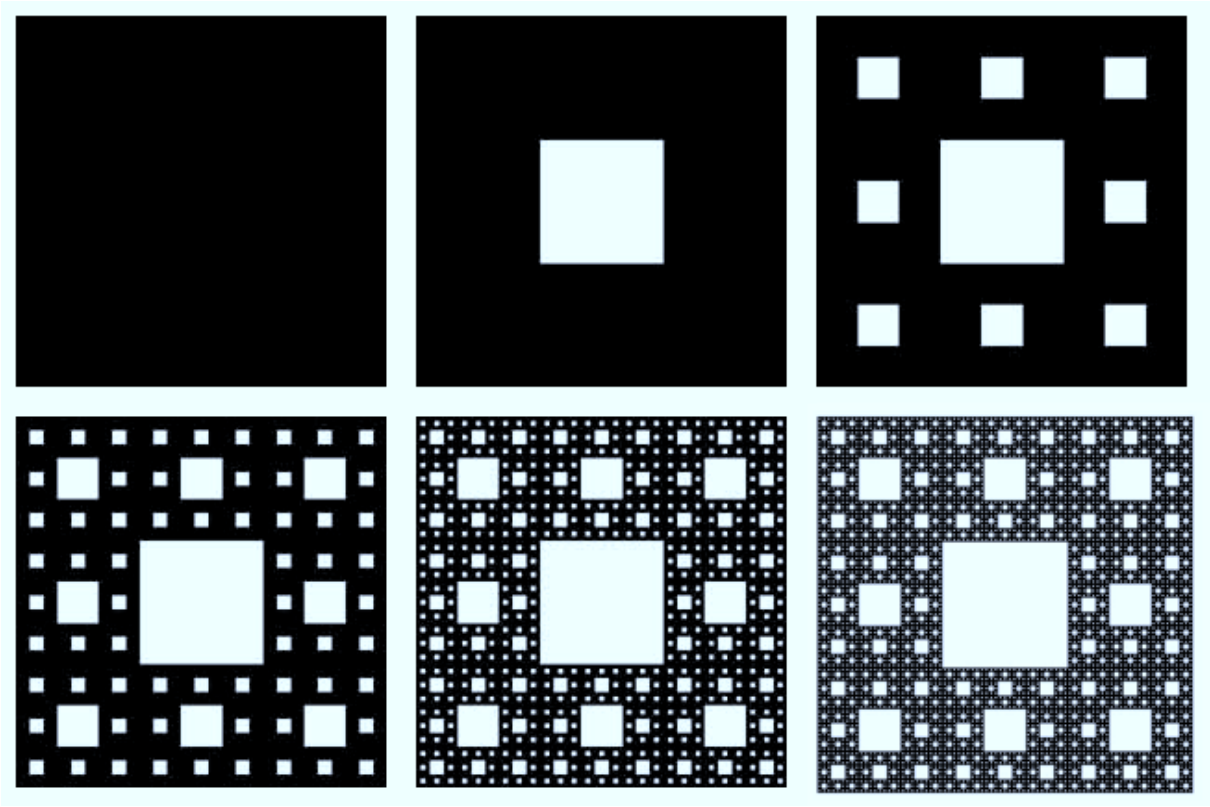
Tento druh fraktálu lze využít při generování textur a přírodních struktur nebo také při odstranění šumu na telekomunikačních linkách. [24]

2.2. Iterační funkční systémy

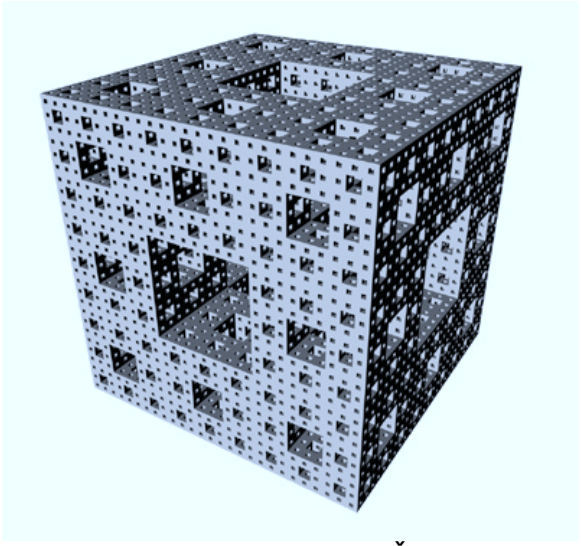
Iterační funkční systém – zkratkou často označován jako IFS, je takový systém, ve kterém funguje jisté geometrické pravidlo. Nejčastěji sem patří plošné obrazce jako čtverec, známým příkladem je například Sierpinského koberec, nebo trojúhelník, příkladem je Sierpinského trojúhelník (viz. Kapitola 3. Sierpinského trojúhelník), ale lze takové pravidlo aplikovat i na tělesa např. krychli. Z krychle utvořený fraktál se pak nazývá Mengerova houba (viz **Obrázek 2.2**). [3]

Příklad je uveden na fraktálu označovaném jako Sierpinského koberec. Čtverec je rozdělen na devět stejných čtverců, a prostřední čtvereček je vynechán. Toto pravidlo bude dále aplikováno na všech osm zbývajících čtverečků. Názorný postup je vyobrazen na **Obrázek 2.1**.

Tyto fraktály našly uplatnění při procedurálním modelování těles. [24]



Obrázek 2.1: Prvních šest iterací Sierpinského koberce. Lze z těchto obrázků názorně pochopit, jak postupně dojdeme k fraktálu [5].



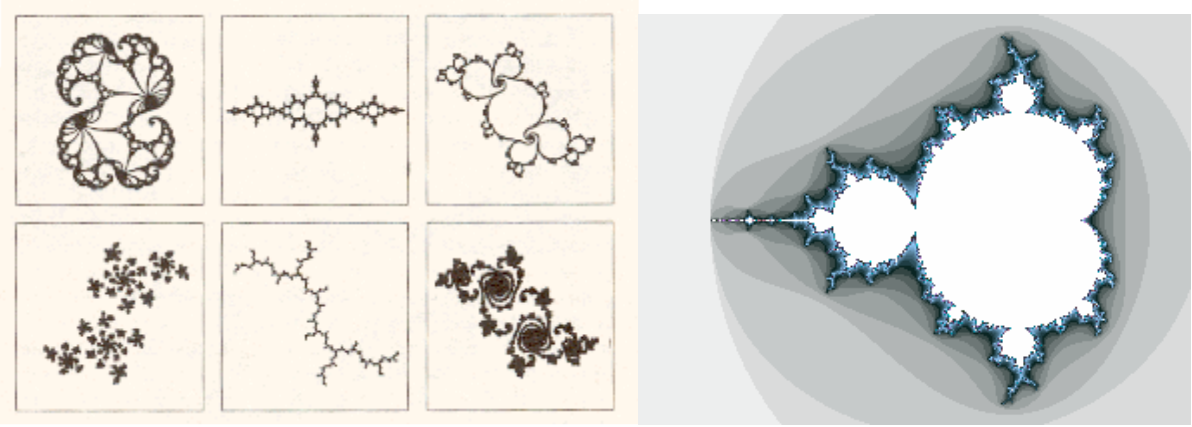
Obrázek 2.2: Mengerova houba. Čtvrtá iterace krychle výše popsaným postupem. Na každé straně krychle vidíme odpovídající iteraci Sierpinského čtverce (koberce) [7].

2.3. Polynomické fraktály

Polynomické fraktály se označují zkratkou TEA, a pokud řeknete heslo „fraktál“ široké veřejnosti, tak naprostá většina si vybaví fraktály právě z této skupiny. Tyto

fraktály nelze většinou využít pro vědecký výzkum, za to z estetického hlediska jsou tyto fraktály pro laiky nejzajímavější. [4]

Už během druhé světové války objevili dva francouzští matematikové G. Julia a P. Fatou pozoruhodné obrazce a dali jim název Juliovy množiny. Na jejich objev navázal v roce 1979 B. Mandelbrot a dokázal najít vztah mezi jednotlivými Julovými množinami a vytvořil, dnes na jeho počest nazývanou, Mandelbrotovu množinu. Mandelbrotova množina je komplexní množina, která v každém svém bodě určuje množinu Juliovu. [3]



Obrázek 2.3: Polynamické fraktály. Vlevo Juliovy množiny, které byly objeveny již během druhé světové války, a vpravo Mandelbrotova množina známá od roku 1979. [6]

Výpočet Mandelbrotovy množiny je vlastně jen zjišťování, zda daný bod z celé komplexní roviny po umocňování konverguje k nule nebo k nekonečnu. V praxi se vezme komplexní číslo, přičte k němu jeho druhou mocninu, to celé se opět umocní a přičte se k tomu opět původní číslo. Pokud výsledek konverguje k nule tak patří do množiny. Vykreslením náležících bodů získáme černobílý obraz Mandelbrotovy množiny. Barvy lze přiřadit podle počtu iterací potřebný ke zjištění konvergence. [3]

2.4. Náhodné fraktály

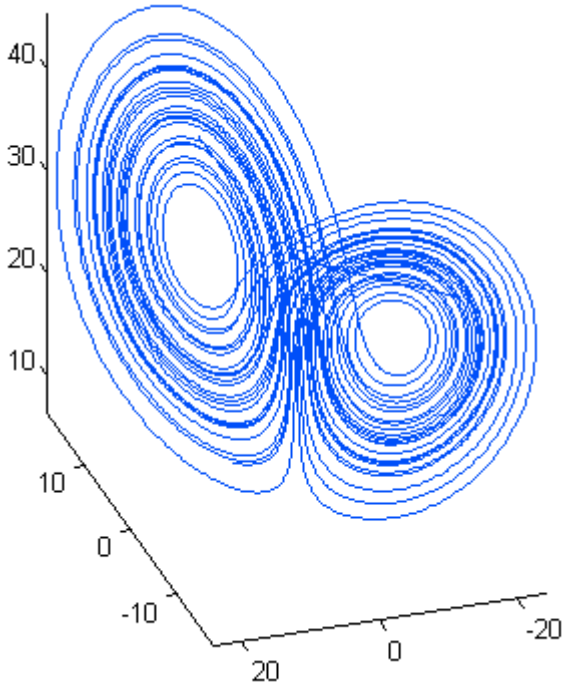
Náhodné fraktály, někdy nazývané chaotické, nejlépe vystihuje Lorenzův atraktor, který ve svém výpočtu zahrnuje i náhodný prvek. Tento atraktor zavedl E. Lorenz v roce 1963 aby mohl popsat chování speciálního vodního kola a jevy v atmosféře. Pro jistou množinu parametrů vykazuje chaotické vlastnosti. Mezi tento typ fraktálů lze zařadit i plošně zachycený Brownův pohyb (lomená čára). [3]

Atraktor je konečný stav systému, zjednodušeně řečeno. U reálného kyvadla je to stav, kdy třením ztratí veškerou svou kinetickou energii, a prostě setrvává na místě. Atraktory lze rozdělit do tří skupin:

- Bodové
- Cyklické (kruhové, osmičkové, …)
- Podivné (chaotické a nekonečné)

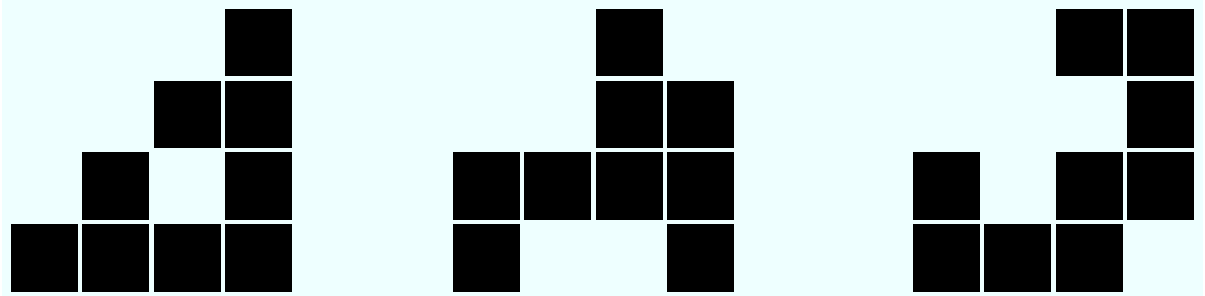
Lorenzův atraktor patří do skupiny podivných atraktorů, protože i z obrázku jde jasně vidět, že se pohybuje po nekonečné křivce. Z tohoto pohledu (viz **Obrázek 2.4**) sice atraktor vypadá jako osmička, ale ve skutečnosti je ve tvaru motýlích křídel zasahující do třech dimenzí – do prostoru. [13]

Tyto fraktály lze použít při generování toků řek nebo modelování turbulencí. Lze je také užít pro popsání vývoje cen na burze. [24]



Obrázek 2.4: Lorenzův atraktor. Je to třírozměrný atraktor, označovaný též jako podivný, s dimenzí 2,6. [6]

Další jednoduchý náhodný fraktál představuje čtverec, na který se aplikuje následující pravidlo. Černý čtverec je rozdělen na čtyři stejné menší čtverce a hornímu levému bude změněna barva na bílou. Toto pravidlo pak lze znova aplikovat na zbylé černé čtverce. Toto pravidlo lze pozměnit tak, aby byl výsledek opravdu náhodný fraktál tak, že bílou barvu přisoudí počítač libovolnému jednomu čtverci. Výsledek pak už nelze předpovědět (viz **Obrázek 2.5**). [9]



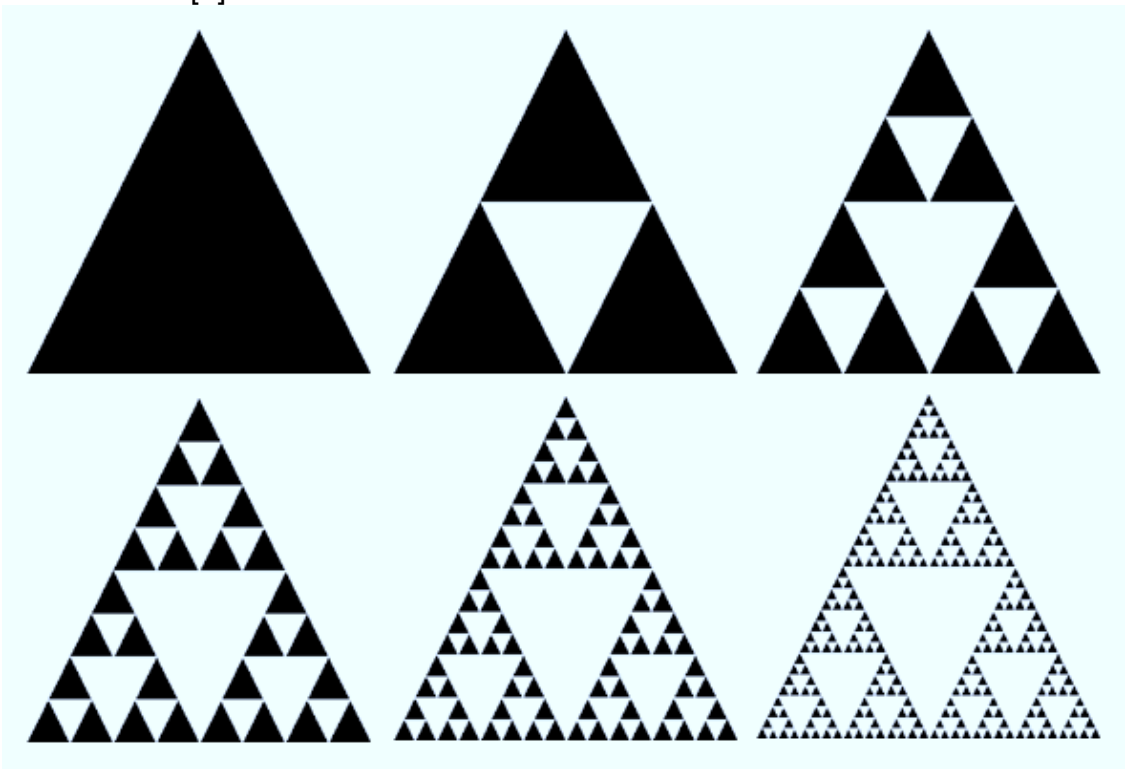
Obrázek 2.5: Náhodný fraktál. Vlevo s pevně přebarveným levým horním čtvercem na bílou, ostatní dva mají upravené pravidlo s náhodným výběrem přebarveného čtverce (možností je více než je uvedeno – slouží jako příklad). [9]

3. SIERPINSKÉHO TROJÚHELNÍK

Sierpinského trojúhelník bude v této kapitole rozebrán podrobněji, protože na něm bude vysvětlena metoda, která pak bude užita na sekvenci bází v DNA (více DNA sekvenci v kapitole 4.2. Struktura DNA a 4.5. Sekvence DNA). Sierpinského trojúhelník je jednoduchý fraktál, jehož konstrukce lze provést dvěmi způsoby – deterministickým (viz kapitola 3.1. Konstrukce trojúhelníku deterministickým způsobem) nebo metodou zvanou Chaos game (více v kapitole 3.2. Konstrukce trojúhelníka metodou chaos game).

3.1. Konstrukce trojúhelníku deterministickým způsobem

Konstrukci Sierpinského trojúhelníka deterministickým způsobem bude vysvětlena na rovnostranném trojúhelníku. Lze použít i obecný trojúhelník, ale nejlépe fraktál vynikne v pravoúhlém nebo právě rovnostranném trojúhelníku. Na počátku máme černý trojúhelník. Spojením středů úseček jeho stran získáme nový trojúhelník, který vybarvíme bíle. Toto pravidlo lze aplikovat znovu na každý černý trojúhelník do nekonečna (viz **Obrázek 3.1**). Popsaná konstrukční metoda je nazývána jako deterministická. [5]



Obrázek 3.1: Konstrukce sierpinského trojúhelníka deterministickým způsobem. Zobrazena nultá až pátá iterace. [5]

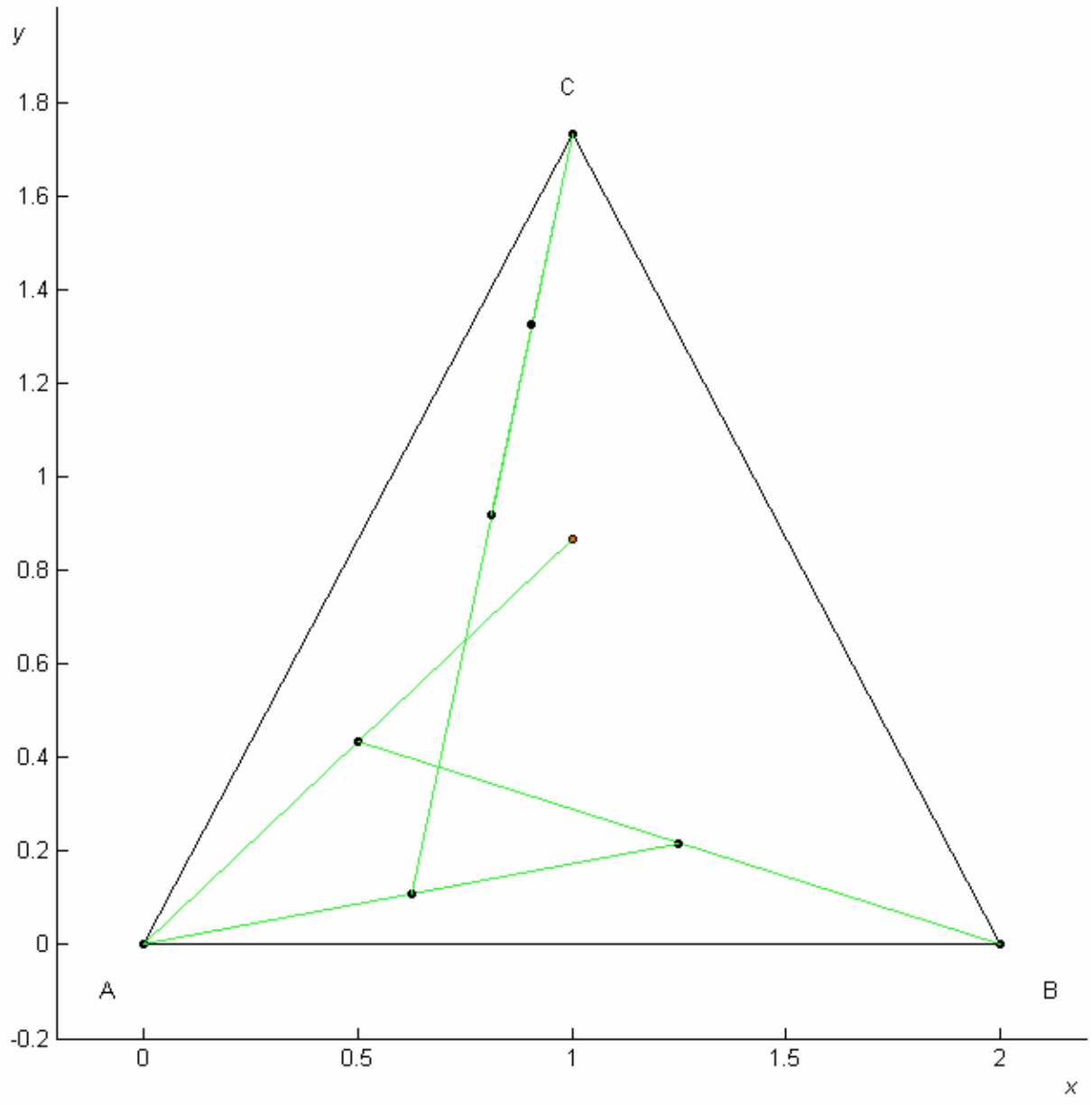
Poznámka autora: Povedlo se mi naprogramovat tyto iterace v prostředí Matlab. Počet iterací je ovšem omezen asi na osm nebo devět, kdy je doba výpočtu ještě přijatelná, to znamená do dvou až tří minut na Intel Centrino Core 2 Duo T7300 s frekvencí 2 GHz (notebook autora – použit pro výpočty). Zdrojový kód je přiložen na konci semestrálního projektu jako Příloha 1. Tento program jsem naprogramoval jen jako ilustraci, pro další výzkum nebude vhodný.

3.2. Konstrukce trojúhelníka metodou chaos game

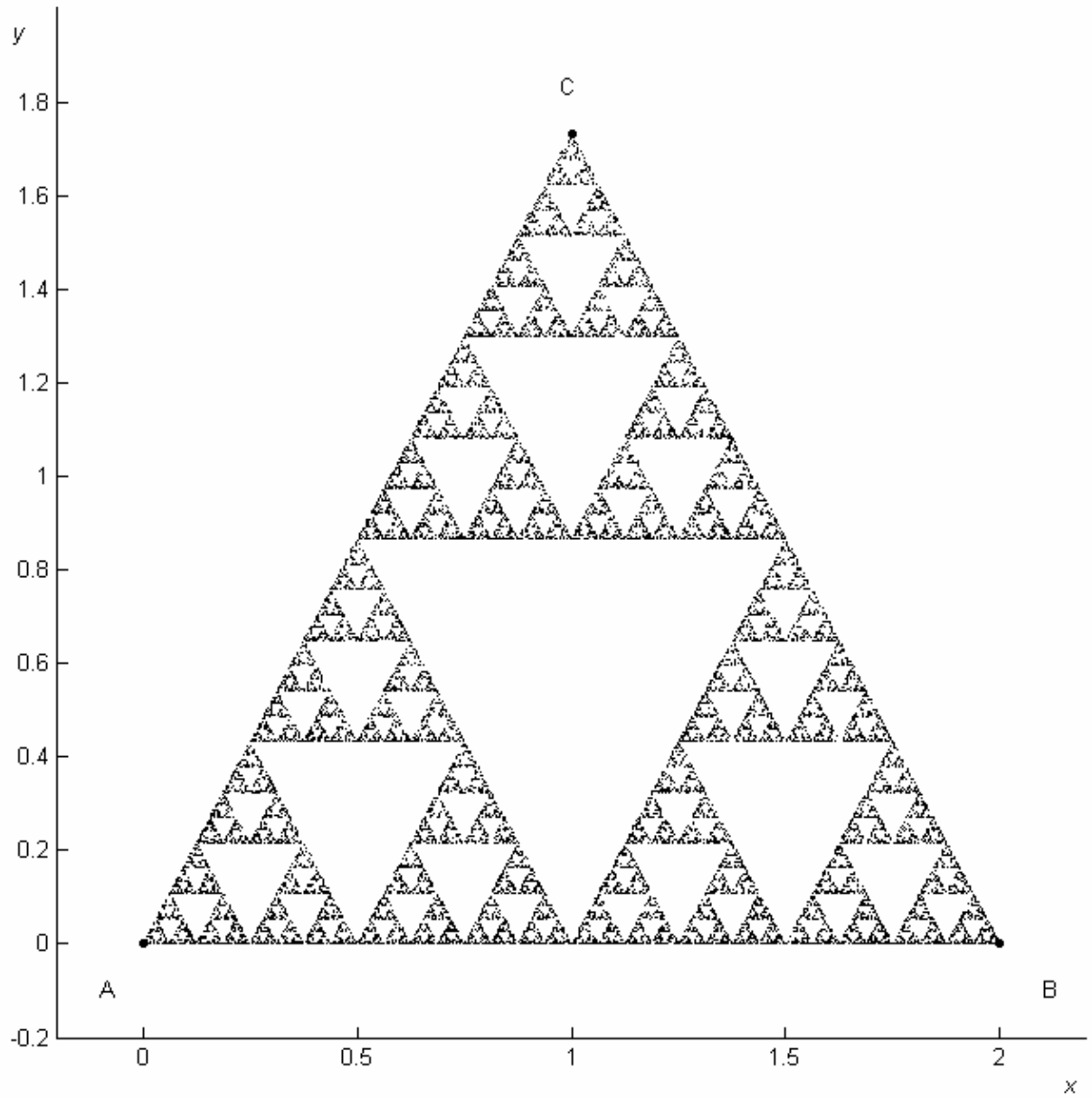
Sierpinského trojúhelník lze vytvořit i metodou chaos game. Metoda je naprosto odlišná od předešlé. Stačí zadat vrcholy trojúhelníka, a popřípadě ještě jeden počáteční bod, který není ovšem nutný, protože jako výchozí bod lze použít libovolný vrchol trojúhelníka. Metoda nechá počítač vybrat vrchol, ke kterému se bude program ubírat. Pomyslným spojem počátečního bodu a zvoleného vrcholu získáme přímku, na které v její polovině nechá vykreslit bod. Tento bod je pak opět vstupem pro další iteraci. [14]

Názorná ukázka (viz **Obrázek 3.2**). V rovnostranném trojúhelníku ABC byl zvolen počáteční bod P, který byl vybarven červeně. Nyní počítač vybral vrchol A, vytvořil spojnici mezi původním bodem a vrcholem A (zelená úsečka) a v polovině vytvořil bod. Tento bod se stává vstupem pro příští iteraci. Opět počítač vybral vrchol, tentokrát B, opět vytvořil spojnici a vykreslil na polovině bod. Při další iteraci zvolil vrchol A, při další C a při páté iteraci opět C.

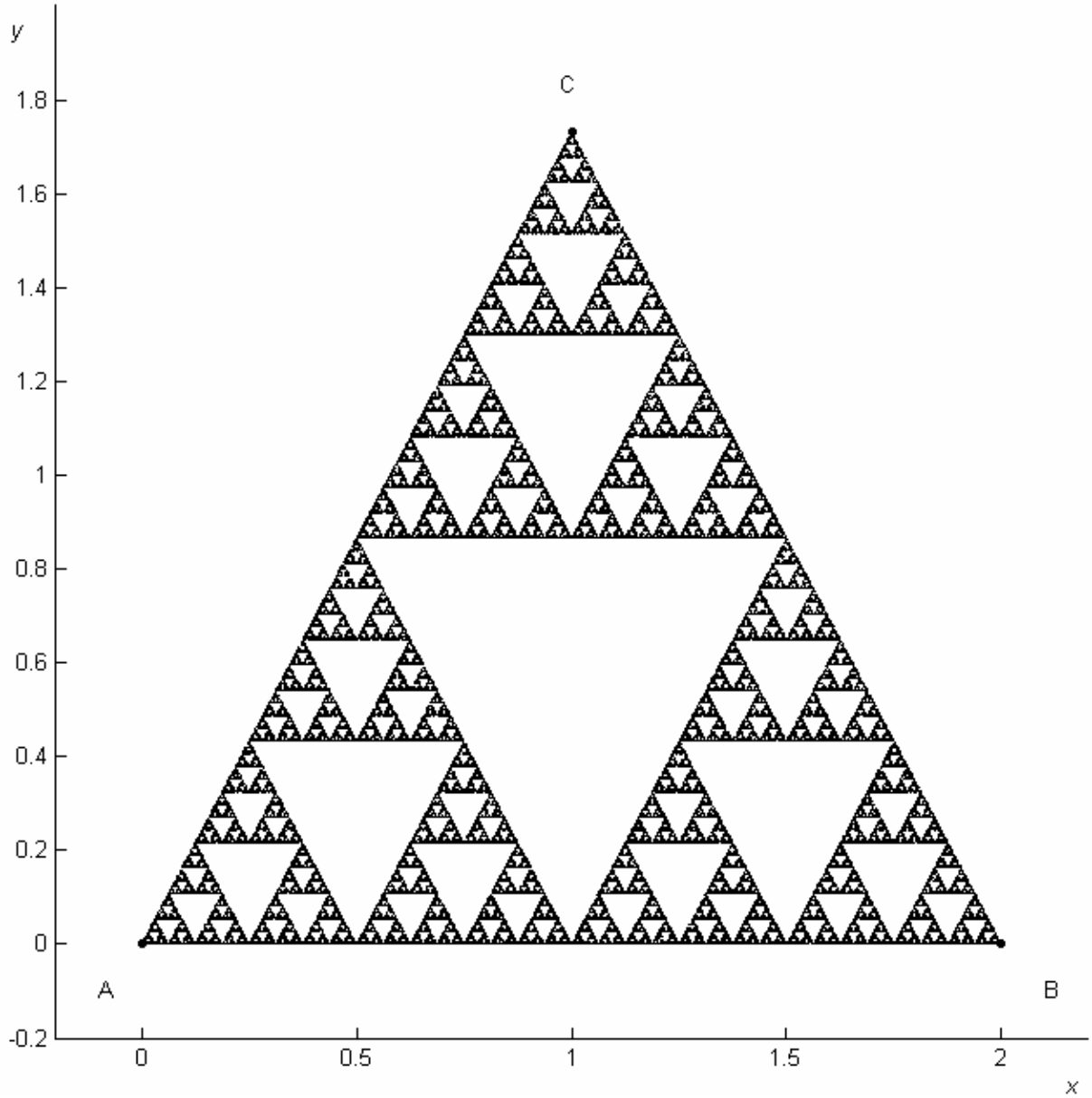
Neustálým opakováním iterací se postupně vykreslí Sierpinského trojúhelník. Neustálé opakování však není praktické, proto je třeba stanovit počet iterací, který bude udávat počet bodů. Všem je asi jasné, že čím více bodů tím více se budeme blížit „dokonalému“ Sierpinského trojúhelníku. Na druhou stranu je zde výpočetní omezení, kdy každý bod navíc potřebuje svůj čas na výpočet. Výpočet čtyřiceti tisíc bodů, nebo méně, je poměrně rychlý a zabere maximálně několik desítek sekund (opět na autorově notebooku; na výkonnějším PC bude výpočet podstatně rychlejší). Výsledný trojúhelník má ovšem plno děr a mezer, které mají být zaplněny. Výpočet sto šedesáti tisíc bodů, nebo více, už ukousne asi minutu až dvě, ale výsledný trojúhelník je o poznání „lepší“. Pro srovnání, trojúhelník o dvaceti tisících bodů je na **Obrázek 3.3**, a byl vypočten asi za osm sekund, zatímco trojúhelník o sto tisících bodů je na **Obrázek 3.4**, a byl vypočten asi za čtyřicet sekund.



Obrázek 3.2: Konstrukce sierpinského trojúhelníka metodou chaos game. Zobrazeno pět iterací. [Generováno vlastním programem v Matlabu]



Obrázek 3.3: Sierpinského trojúhelník konstruovaný metodou chaos game po 20 000 iterací a byl vypočten asi za 8 sekund. [Generován vlastním programem v Matlabu]



Obrázek 3.4: Sierpinského trojúhelník konstruovaný metodou chaos game po 100 000 iterací a byl vypočten asi za 40 sekund. [Generován vlastním programem v Matlabu]

4. SEZNÁMENÍ S DNA

Celá genetická výbava jedince, nazývaná též **genom**, je rozdělena do **chromozomů**. Každý chromozóm obsahuje několik **genů** (více v kap. 4.5. Sekvence DNA) a geny pak tvoří DNA řetězec, nebo dvoušroubovici. [19]

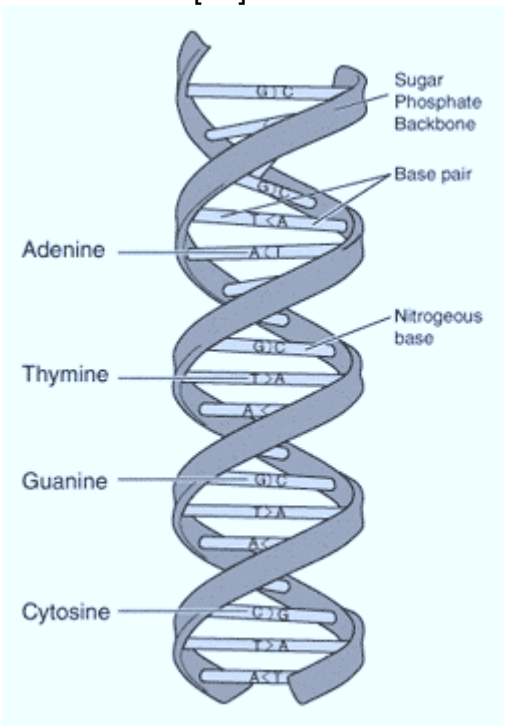
DNA neboli deoxyribonukleová kyselina (česky někdy označována jako DNK) je součástí každé živé buňky. Jednodušší RNA postačí jednoduchým virům a bakteriím, nebo jako „pomocník" při dělení buňky a přenosu částí DNA. Nukleové kyseliny jsou vysokomolekulární látky, tvořené nukleotidy (viz kapitola 4.2. Struktura DNA) jako základní stavební kameny DNA. Daná buňka si z tohoto množství dat vybere vždy jen sobě potřebnou část, a s tou nadále pracuje. DNA se používá k syntéze proteinů v buňce, nebo slouží k přenosu genů do nových (dceřinných) buněk při dělení – tzv. replikace (viz 4.3. Replikace DNA). [10,15]

4.1. Historie výzkumu DNA a DNA sekvence

DNA strukturu objevili v James D. Watson a F. H. C. Crick v roce 1953, kteří za tento objev dostali společně s M. Wilkinsnem Nobelovu cenu za fyziologii a medicínu v roce 1962. V letech 1972 až 1975 F. Stanger a W. Gilbert vyvinuli techniku sekvencování DNA. [10]

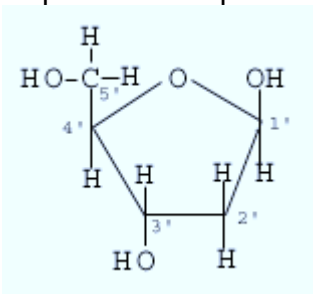
4.2. Struktura DNA

Základní strukturu DNA je zobrazena na obrázku níže (viz **Obrázek 4.1**). Význam anglických popisků - vlevo anglické názvy bází A, T, G, C a vpravo shora Sugar Phosphate Backbone = Cukrofosfátová páteř, Base pair = komplementární pár a Nitrogeous base = nukleotidová báze. [11]

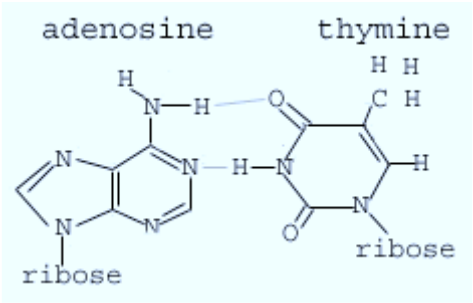


Obrázek 4.1: Dvoušroubovice DNA. [11]

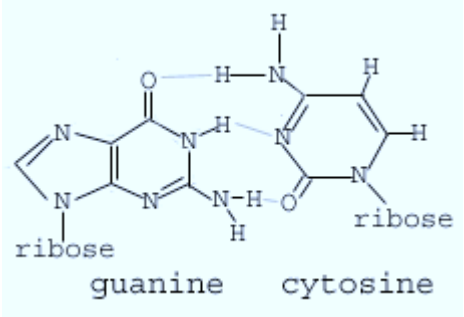
DNA je uchovávána v chromozómech v jádru buňky zpravidla v podobě pravotočivé dvoušroubovice o průměru 1,9 nm, ale část lze najít i v mitochondriích. Ta je tvořena fosfátovou páteří, přesněji kyselina trihydrogenfosforečná, a přes cukr (ribóza pro RNA, deoxyribóza pro DNA; struktura viz **Obrázek 4.2**) je navázána dusíkatá báze. Zbytek kyseliny fosforečné, cukr a báze je označována pojmem **nukleotid**. Báze jsou čtyři a dvou druhů – purinové (A = adenin, G = guanin) a pyrimidinové (C = cytosin, T = thymin). Dvě báze tvoří **komplementární pár**, označován jako **bp** (anglicky base pair), kdy se pojí jedna báze purinová a jedna pyrimidinová, a pomocí vodíkových můstků drží obě ramena dvoušroubovice DNA pospolu. Báze se k sobě nepojí náhodně, ale vždy adenin s thyminem nebo thymin s adeninem (A-T, T-A), spojené přes dvě vodíkové vazby (viz **Obrázek 4.3:** Vazba adenin - thymin a jejich strukturní vzorce. Na obrázku jsou jasně vidět dvě vodíkové vazby. [12]), a cytosin s guaninem nebo guanin s cytosinem (C-G, G-C), spojené přes tři vodíkové vazby (viz **Obrázek 4.4**). Mezi pyrimidinové se někdy přidává ještě U = uracil (není součástí DNA, ale v těle se vyskytuje jako báze v RNA a nahrazuje thymin – to znamená, že tvoří komplementární pár s adeninem). [10, 15]



Obrázek 4.2: Strukturní vzorec cukru deoxyribózy. V cyklu jsou popsány čísly uhlíky, na které je možno navázat bázi. [12]



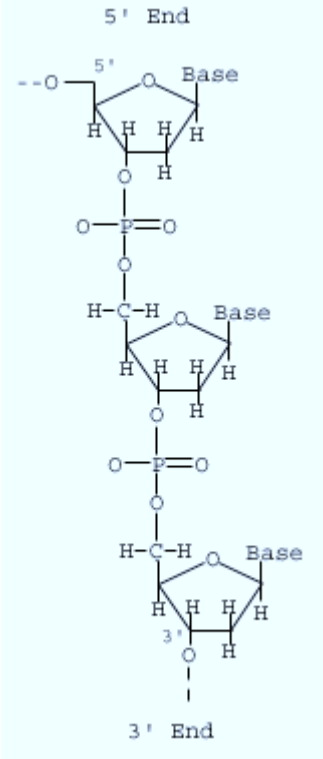
Obrázek 4.3: Vazba adenin - thymin a jejich strukturní vzorce. Na obrázku jsou jasně vidět dvě vodíkové vazby. [12]



Obrázek 4.4: Vazba guanin - cytosin a jejich strukturní vzorce. Na obrázku jsou patrné tři vodíkové vazby. [12]

Nukleotidy se spojují přes diesterické vazby do řetězců …-cukr-fosfát-cukr-fosfát-cukr-… a tvoří polynukleotidový řetězec (viz **Obrázek 4.5**). [15]

Spojením této páteře s komplementárním párem bází, která na svůj druhý konce naváže druhou páteř, dostaneme úplný strukturní vzorec části DNA. Byl by značně komplikovaný a nepřehledný, proto se znázorňuje schematicky tak, jak je to naznačeno na **Obrázek 4.1**. [15]



Obrázek 4.5: Struktura DNA páteře posloupnosti cukr-fosfát-cukr. Označení uhlíků na počátku a konci řetězce 5' a 3' (čteno: pět konec a tři konec) a místa, kde je navázána báze je označena heslem **Base**. [16]

4.3. Replikace DNA

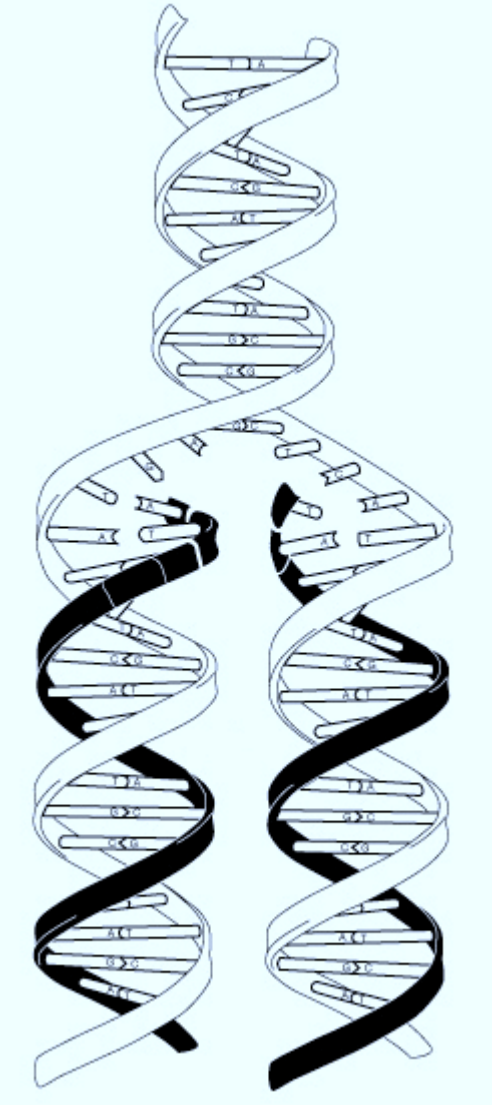
Replikace DNA (česky také překládáno jako samozdvojení) je základní proces při dělení celé buňky a slouží k přenosu genetického materiálu z jedné matečné buňky na dvě dceřinné se stejnou genetickou výbavou. Pořadí nukleotidů (tzv. primární struktura) má základní význam pro přenos genetické informace, proto se pořadí nukleotidů při dělení nesmí zaměnit. [15]

Na počátku replikace se účinkem enzymu DNA-polymeráza rozvolní vodíkové vazby mezi bázemi. Takto rozvolněné části DNA řetězce pak slouží jako **matrice** (vzory) k přiřazení volných komplementárních párů. Vzniknou tak dva DNA řetězce, ve kterých je polovina z matečné buňky a polovina komplementárně dodaná (viz **Obrázek 4.6**). Z toho lze vypozorovat, že pro přenos genetické informace stačí jeden řetězec dvoušroubovice DNA. [15]

Ve skutečnosti je celý proces dělení složitější. Procesu replikace se účastní RNA (ribonukleová kyselina), která je podstatně menší než DNA (tvoří ji řádově desítky až stovky nukleotidů), a při procesu dělení plní různé funkce. RNA rozlišujeme tři základní typy (podle toho, kde se nachází, nebo jakou plní funkci):

- ribozomální rRNA
  - stavební funkce v ribozomu (jedna z buněčných organel)

- mediátorovou mRNA
  - získává komplementární doplňky částí DNA přímo v jádru buňky; následně je uvolněna do cytoplazmy buňky (roztok vyplňující buňku a prostor mezi organelami)
- transferovou tRNA
  - zajišťuje přesun (transport) aminokyselin k ribozomu.



Obrázek 4.6: Replikace DNA. Původní dvoušroubovice se rozdělí a komplementací se přidají nové řetězce (černě). Vznikají dvě totožné dvoušroubovice. [18]

4.4. Genetická informace

Genetická informace je dána posloupností nukleotidů, tedy sekvencí **genů** (více v kap. 4.5. Sekvence DNA). Sekvenci lze pak rozdělit na **kodóny,** neboli triplety, což je posloupnost tří po sobě jdoucích nukleotidů, a ta představuje právě jednu aminokyselinu. **Gen**, neboli vloha, je jednotka zodpovědná za vznik dědičné vlastnosti (rozebráno na konci kapitoly). [15]

Triplety kódují aminokyseliny, které se syntetizují v buňce. Počet syntetizovaných bílkovin je asi dvacet a počet různých kombinací bází je

$$4^3 = 64 \text{ kodonů.} \qquad (11)$$

Jasný nepoměr mezi možnostmi kodonů (počet v rovnici (11)) a skutečným počtem syntetizovaných bílkovin (dříve řečeno – asi 20) říká, že danou aminokyselinu (tj. bílkovinu) může kodón kódovat několika různými kombinacemi (kombinace RNA viz **Tabulka 4**). Šedě jsou vybarveny buňky START a STOP, které značí kombinace bází určující počátek a konec sekvence pro syntézu bílkovin. [15]

Tabulka 4: Přehled kodonů a aminokyselin pro RNA. Oproti DNA je zde místo thyminu (T) uracil (U). [inspirace ve zdroji 17; přepsáno]

První písmeno kodonu		Druhé písmeno kodonu								Třetí písmeno kodonu
		U		C		A		G		
U	UUU	Fenylalanin	UCU	Serin	UAU	Tyrosin	UGU	Cystein	U	
	UUC		UCC		UAC		UGC		C	
	UUA	Leucin	UCA		UAA	STOP	UGA	STOP	A	
	UUG		UCG		UAG		UGG	Tryptofan	G	
C	CUU	Leucin	CCU	Prolin	CAU	Histidin	CGU	Arginin	U	
	CUC		CCC		CAC		CGC		C	
	CUA		CCA		CAA	Glutamin	CGA		A	
	CUG		CCG		CAG		CGG		G	
A	AUU	Izoleucin	ACU	Treonin	AAU	Asparagin	AGU	Serin	U	
	AUC		ACC		AAC		AGC		C	
	AUA		ACA		AAA	Lysin	AGA	Arginin	A	
	AUG	Methionin,START	ACG		AAG		AGG		G	
G	GUU	Valin	GCU	Alanin	GAU	Kys. asparágová	GGU	Glycin	U	
	GUC		GCC		GAC		GGC		C	
	GUA		GCA		GAA	Kys. glutamová	GGA		A	
	GUG		GCG		GAG		GGG		G	

Transkripce (česky přepis) genů se částečně podobá replikaci. Na počátku se opět oba řetězce DNA oddálí a ke kodonům uložených v DNA se najde komplementární molekula mRNA. mRNA molekuly se za sebou spojují a získá se komplementární vlákno k DNA. Toto vlákno se uvolní z jádra a dále se zpracovává v ribozomech (další buněčná organela). Zde dochází k **translaci**. Translace (česky překlad) je děj na ribozomech, kde se ke kodonům mRNA najdou **antikodony** (komplementární aminokyseliny) tRNA. Molekuly tRNA se spojí do polypeptidu a jsou následně uvolněny do cytoplazmy buňky (hmota vyplňující buňku a prostor mezi organelami). [15]

4.5. Sekvence DNA

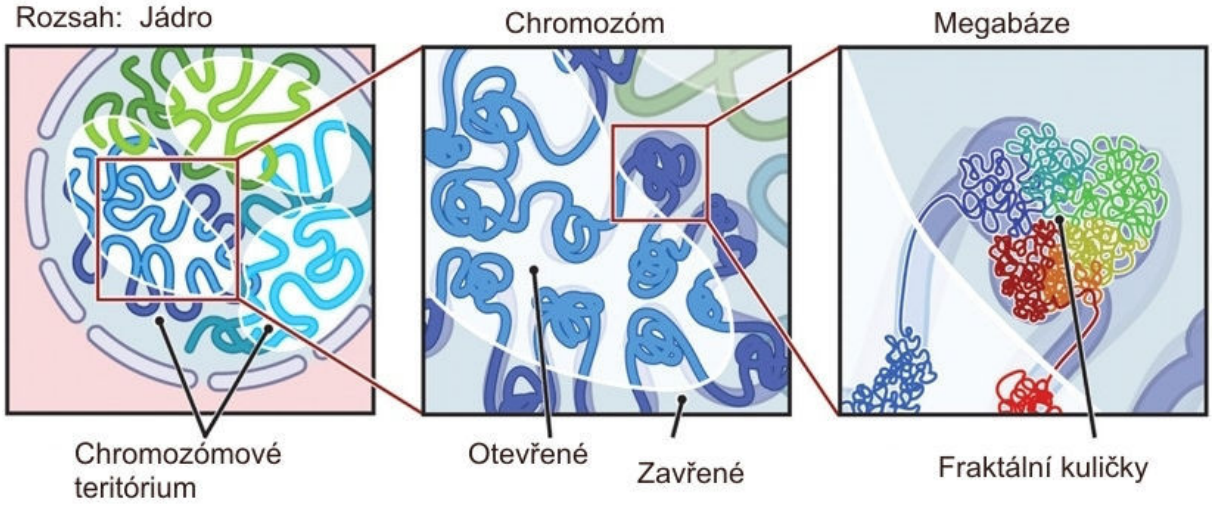
Gen je v DNA sekvenci rozdělen do několika úseků – exonů (nesoucí informaci) a úseků oddělujících – intronů. Přesná funkce intronů není doposud známá. Spekuluje se o jisté kontrolní funkci, kdy má kontrolovat informaci uloženou v exonech, nebo hraje jistou úlohu při evoluci, kdy nové vlastnosti genů mohou vznikat kombinací exonů, nebo má určitou ochranou funkci, kdy chrání informaci uloženou v exonech. Intron je při transkripci genu odebrán, aby mRNA nesla jen užitečnou informaci. [15]

5. FRAKTÁLNÍ STRUKTURY V DNA

Fraktální struktury v DNA jsou již dnes známy, budou v této kapitole přiblíženy.

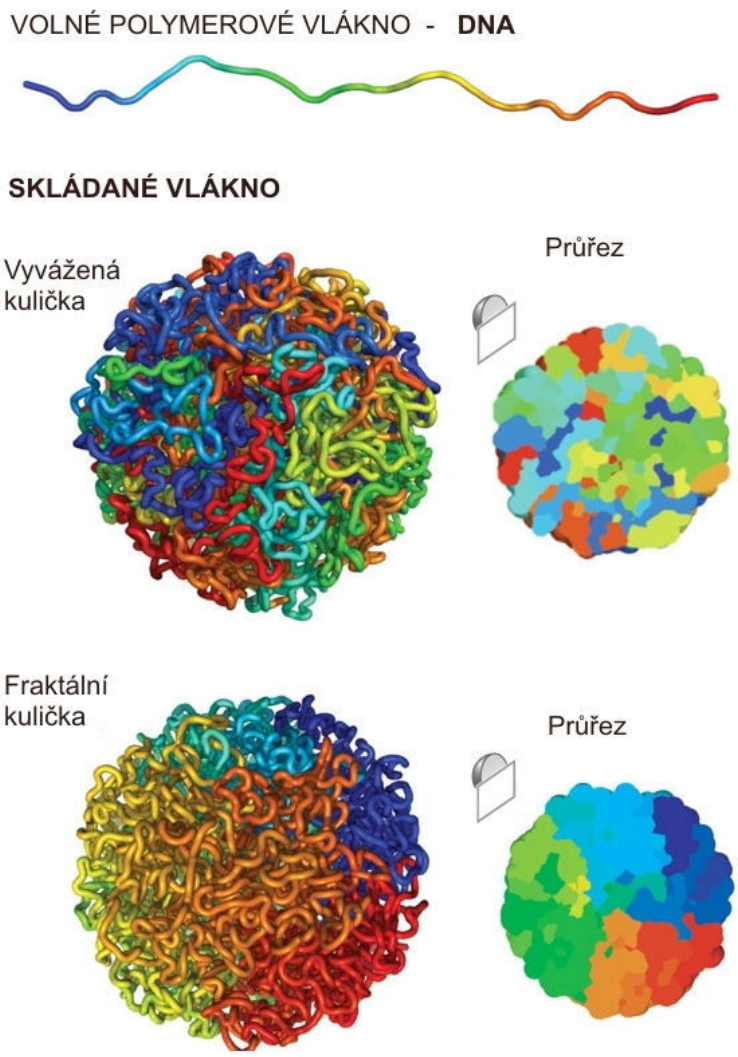
5.1. Fraktální globule

Jak bylo již dříve zmíněno, DNA je uchována v chromozomech, a jsou naskládány v jádru buňky (viz kap. 4. Seznámení s DNA). Kdyby se DNA uchovávaná v jedné buňce rozvinula, tak by dosahovala délky okolo dvou metrů. Jak se ovšem takto dlouhá DNA vejde do jediného jádra buňky? Každý asi přijde na to, že ji lze poskládat. Je zde ovšem problém, že DNA se stále používá k syntéze bílkovin, a tak používané části by měly být stále na dosah. Tento problém řeší fraktální gloubule (zobrazena na obrázku **Obrázek 5.1**). [20]



Obrázek 5.1.: Fraktální globule v jádru buňky. [20]

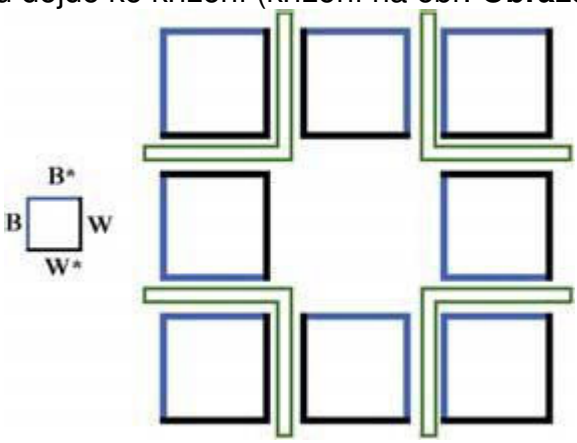
Fraktální globule sdružuje u sebe chromozómy, nebo jejich části, označované jako lokusy, podobných vlastností. To, že se jedná o fraktál, dokázali američtí vědci z Harvardské University, kdy danou fraktální kuličku (globuli) podrobili zkoumání. „Ztuhlou“ globuli ve formaldehydu podrobili štěpení enzymem a globule se rozpadla na malé kousky. Tyto kousky pak nechali počítačem složit zpět do globule jako velké 3-D puzzle. Pro potvrzení pak nechali na globuli působit jiný enzym, ta se rozpadla na jiné kousky, které po poskládání dohromady, dostali stejnou kuličku. Detail fraktální kuličky je na obrázku **Obrázek 5.2**. Na tomto obrázku je také srovnání s kuličkou, která byla uvažována dříve – skládaná a vyvážená. Por srovnání je zde také zobrazen řez kuličkami. [20]



Obrázek 5.2.: Detail fraktální kuličky. [20]

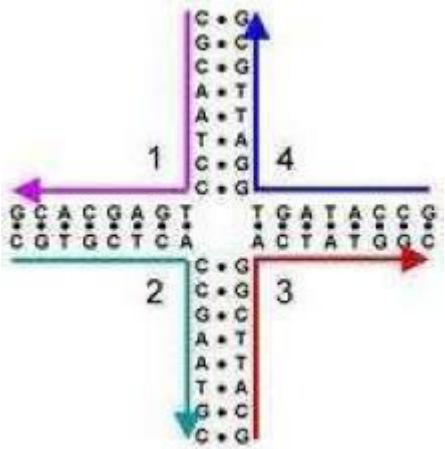
5.2. 2-D nanostruktura DNA

Nanostruktury DNA se odvozují od Sierpinského koberce (viz. **Obrázek 2.1**). Čtverec je rozdělen na části podle obrázku níže (**Obrázek 5.3**). Takto vzniklé čtverce lze pak považovat za stavební čtverce nanostruktur, kdy každou stranu čtverce tvoří dvojvlákno DNA a rohu dojde ke křížení (křížení na obr. **Obrázek 5.4**). [22]



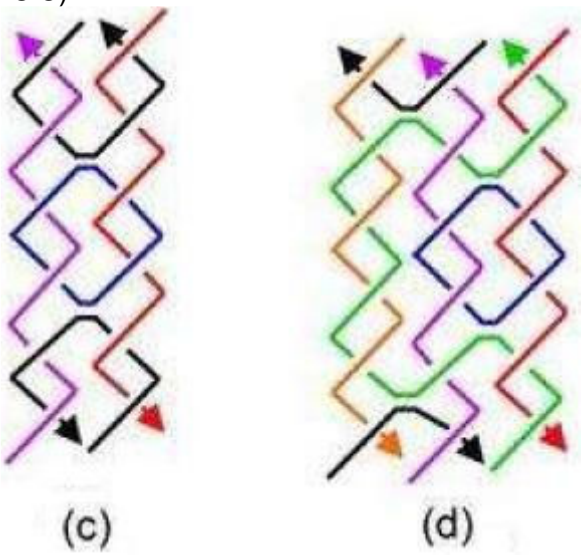
Obrázek 5.3.: Dělení Sierpinského koberce. [22]

Strany čtverce jsou označeny písmeny B, B*, W a W*, kde vlákna bez hvězdiček (tj. B a W) se stočí po směru hodinových ručiček, a vlákna s hvězdičkou (tj. B* a W*) jsou průběžná. Dvojvlákno DNA je ovšem tvořeno dvěmi vlákny, která lze rozdělit, a vést je odděleně (rozdělené křížení je rozkresleno na obrázku **Obrázek 5.4**). Dvojvlákno je děleno v křížení tak, aby stabilizovalo tuto pozici. Bez tohoto křížení by se dvojvlákno samo stočilo do podoby dvoušroubovice. [22]



Obrázek 5.4.: Detail křížení čtyř vláken v rozích čtverců. Čísly jsou označeny řetězce. [24]

Vlákna v těchto strukturách mohou podstupovat dvojí (double crossover = DX) nebo dokonce trojí křížení (triple crossover = TX), obojí je zachyceno na obrázku níže (viz obr. **Obrázek 5.5**).



Obrázek 5.5.: Dvojité a trojité křížení DNA vláken. DX na (c) a TX na (d) části. [25]

5.3. Struktura XOR

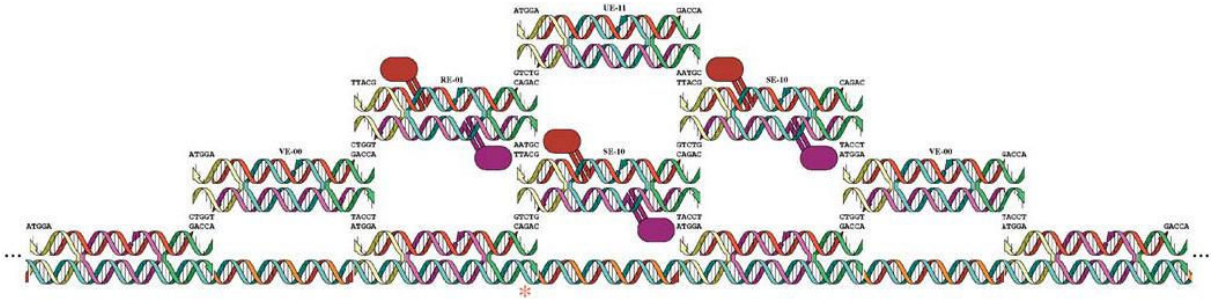
2D struktura může plnit jednoduchou funkci XOR. Pravdivostní tabulka je uvedena dále (tab. **Tabulka 5:** Pravdivostní tabulka). Strukturu můžeme rozdělit do 4 bloků, v závislosti na vstupních proměnných, podle pravdivostní tabulky (obr. **Obrázek 5.7Obrázek 5.7**). **Obrázek 5.7** A ukazuje postupný výpočet funkce XOR, a po několika iteracích je vidět, že struktura jedniček vykreslí Sierpinského trojúhelník. Na

B je ukázka bloků a definování vstupů dole a výstupů nahoře každého bloku. Na C jsou vypsány všechny možnosti bloků a jejich skládání do struktury. Hvězdičkou je označeno místo inicializace – první jednička pro zahájení výpočtu. D již ukazuje plošnou strukturu tohoto výpočtu bez chyb. E ukazuje výpočet pro více inicializačních míst (hvězdička) a vzniklé chyby ve struktuře (buňky s křížkem). [27]

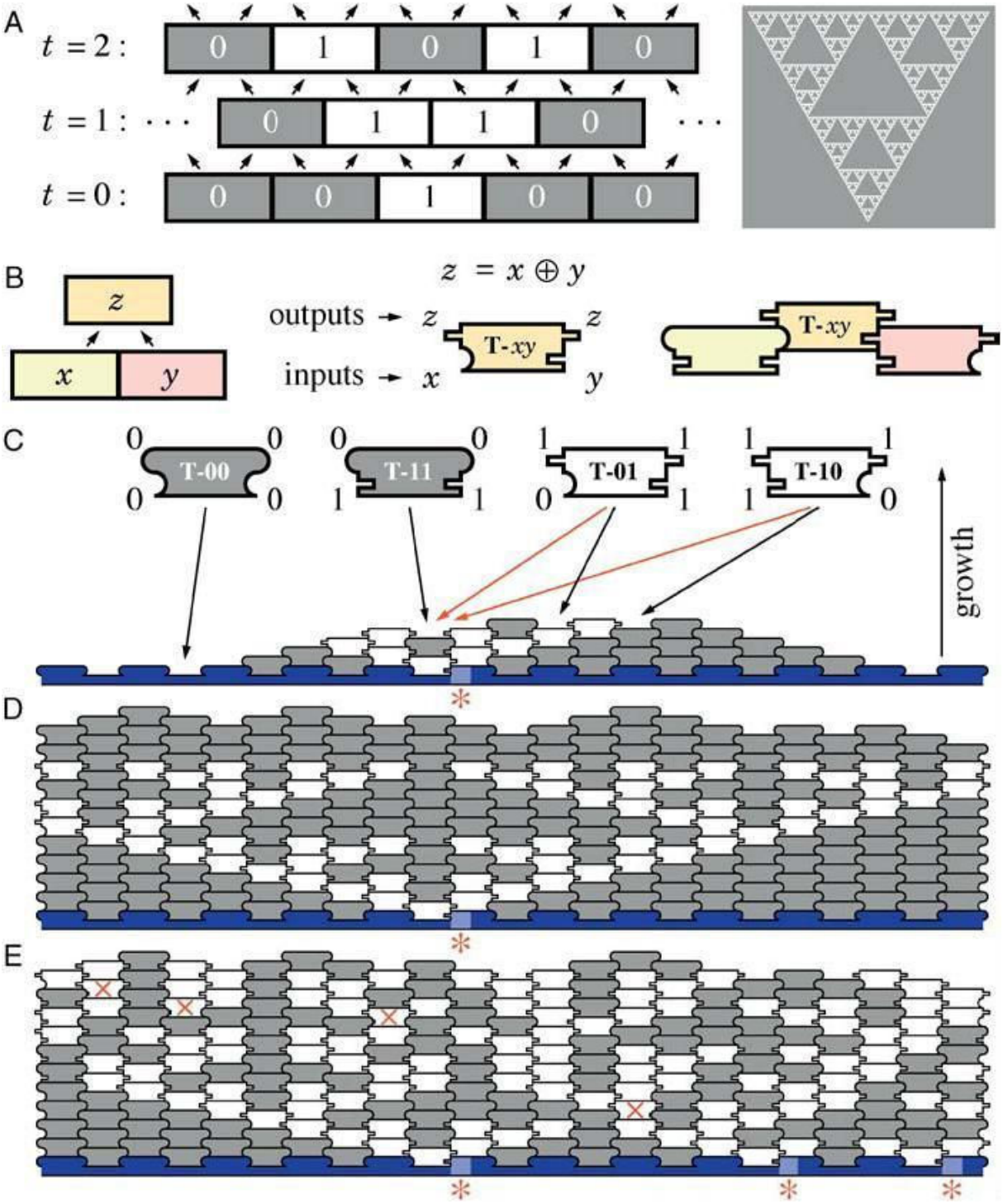
Struktura v DNA sekvencích je zachycena na obr. **Obrázek 5.6**. Jelikož tato struktura plní matematickou operaci, lze ji pak použít pro binární výpočty, a z nich lze vytvořit DNA počítače. [24]

Tabulka 5: Pravdivostní tabulka XOR. [24]

a	b	XOR
0	0	0
0	1	1
1	0	1
1	1	0



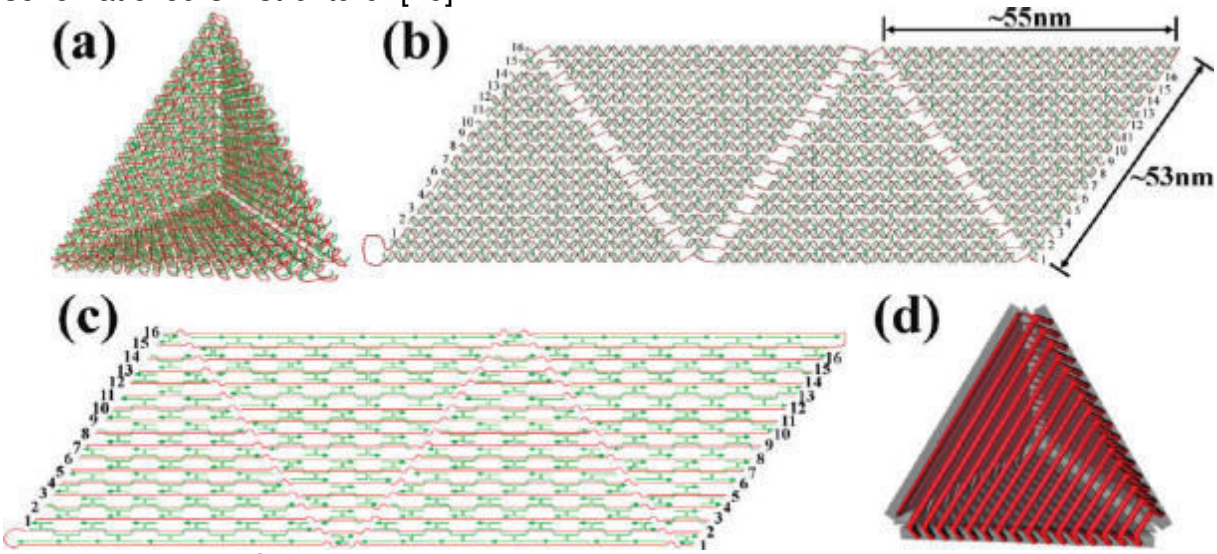
Obrázek 5.6.: Struktura XOR v DNA sekvencích. [27]



Obrázek 5.7.: Struktura funkce XOR. [27]

5.4. 3-D nanostruktura DNA

Podobně jako 2-D struktura je odvozena od Sierpinského koberce tak i 3-D struktura je odvozena od tohoto prostorového uspořádání do tzv. Mengerovy houby (viz. **Obrázek 2.2**). Tato struktura pak v prostoru vyplňuje část prostoru tvořenou třemi rovnostrannými trojúhelníky = čtyřstěn = tetrahedron (zobrazen na obr. **Obrázek 5.8**). Obrázek ukazuje strukturu tetrahedronu v prostoru (a), rozvinutou strukturu (b, c), kde je ještě zachycena interakce mezi dvoušroubovicemi a (d) znázorňuje schématickou 3D strukturu. [26]



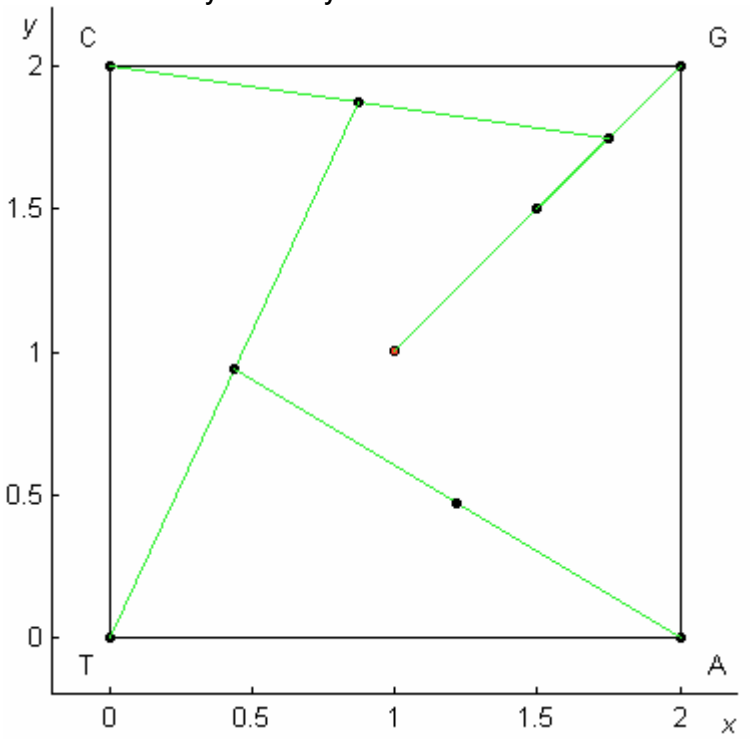
Obrázek 5.8.: Tetrahedron. [26]

6. ZKOUMÁNÍ SEKVENCE DNA

Sekvence DNA (popsáno v kap. 4. Seznámení s DNA a 4.5. Sekvence DNA) bude vložena na vstup upravené metody chaos game v Sierpinského trojúhelníku (popsáno v kap. 3.2. Konstrukce trojúhelníka metodou chaos game).

6.1. Úprava metody chaos game pro zkoumání sekvence DNA

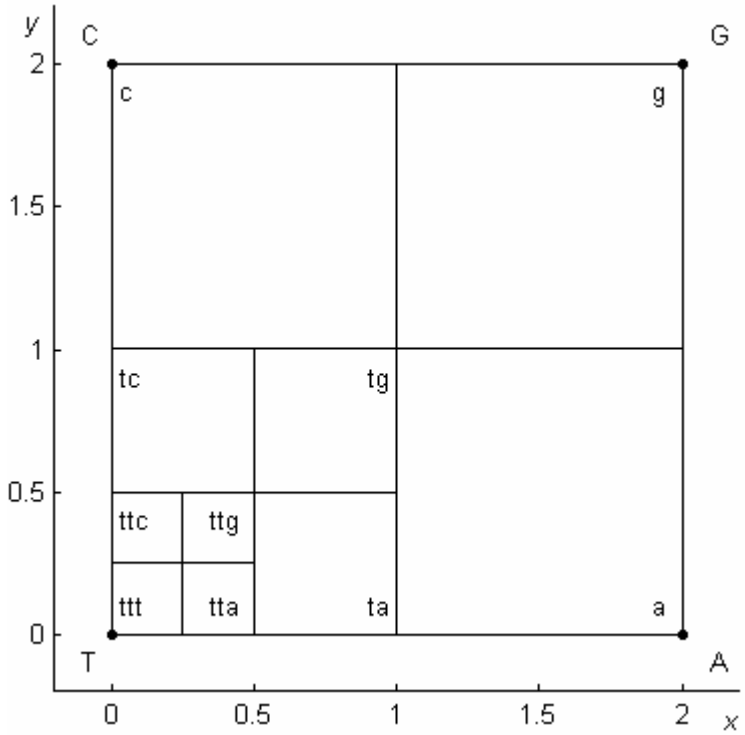
Sierpinského trojúhelník bude rozšířen o další vrchol a upraven na čtverec o vrcholech ACGT (zkratky DNA bází; podrobnosti v kap. 4.2. Struktura DNA). Popis vrcholů převzat ze zdroje [28]. Nyní už výběr vrcholu nebude volen počítačem, ale bude pevně dán posloupností bází, která je na vstupu programu. Na obrázku níže (**Obrázek 6.1**) je zobrazena ukázka DNA kódu pro posloupnost GGAGT. Počáteční bod uprostřed čtverce – červeně, spojnice půdního bodu a vrcholu, který je určen posloupností bází – zeleně a vykreslený bod – černě.



Obrázek 6.1.: Ukázka metody chaos game ve čtverci ACGT. [Generováno vlastním programem v Matlabu]

6.2. Segmentování čtverce ACGT

Čtverec ACGT lze rozdělit na menší dílčí čtverce a popisují se podle vrcholu, ke kterému mají nejblíže. Čtverec rozdělíme na čtyři menší spojením protilehlých středů stran. Menší čtverce se popisují A, C, T a G. I tyto menší čtverce lze opět rozdělit na čtyři menší. Ty se popisují nejdříve písmenem většího čtverce, a pak následně menšího. Příklady popisu: AA, AC, TT, GC… Tento postup zase lze zopakovat i na menší. Popis pak bude tří-písmenný (viz **Obrázek 6.2**). Třípísmenný popis pak odpovídá posloupnosti znaků v kodónu a malý čtverec pak reprezentuje daný kodón.

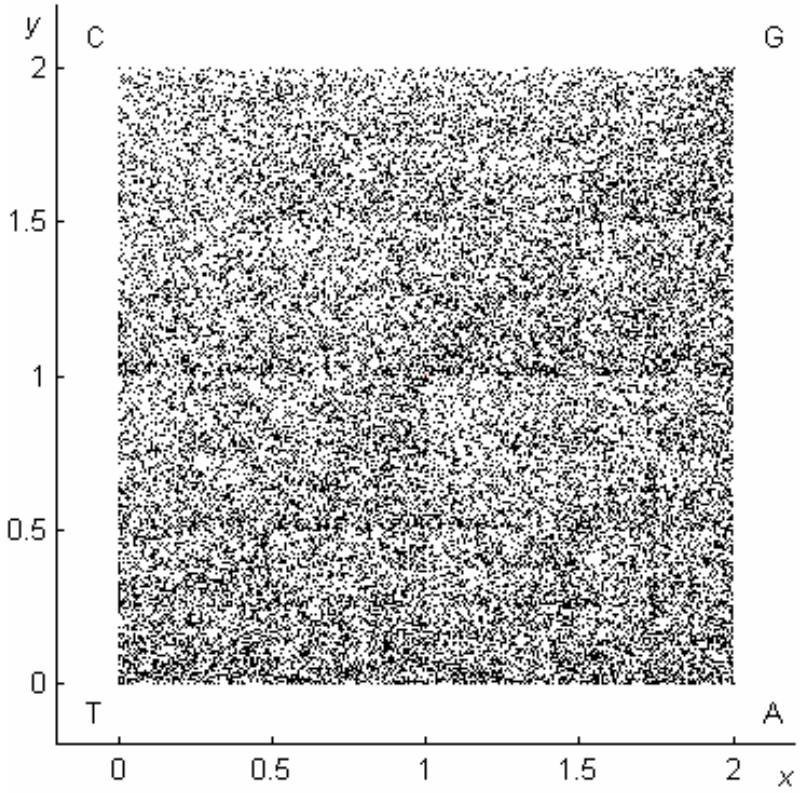


Obrázek 6.2.: Segmentace čtverce ACGT a popis čtverců podle vrcholu, ke kterému naléhají. [Generováno vlastním programem v Matlabu]

7. ZPRACOVÁNÍ SEKVENCÍ

7.1. HEXA hexosaminidáza A

Z veřejné databáze [23] byla získána genetická informace této bílkoviny a uložena do txt souboru. Funkce *dna2ctver* poté načítá sekvenci nukleotidů z tohoto souboru. Výsledný obrázek získaný funkcí je uveden níže na obrázku níže (**Obrázek 7.1**). Vstupní proměnné funkce jsou *stran*, která zapíná možnost vykreslení obvodových stran čtverce ACGT (1 = zapne; 0 = default), *mark*, která určí velikost vykreslovaných bodů (doporučuji: 1,2,3; default = 1) a *prod*, která vnese prodlevu do vykreslování a lze jej pak pozorovat v reálném čas (doporučené hodnoty: 0,001 – 0,000 001; default = 0). Zadávají se v pořadí *dna2ctver(stran, mark, prod)*.



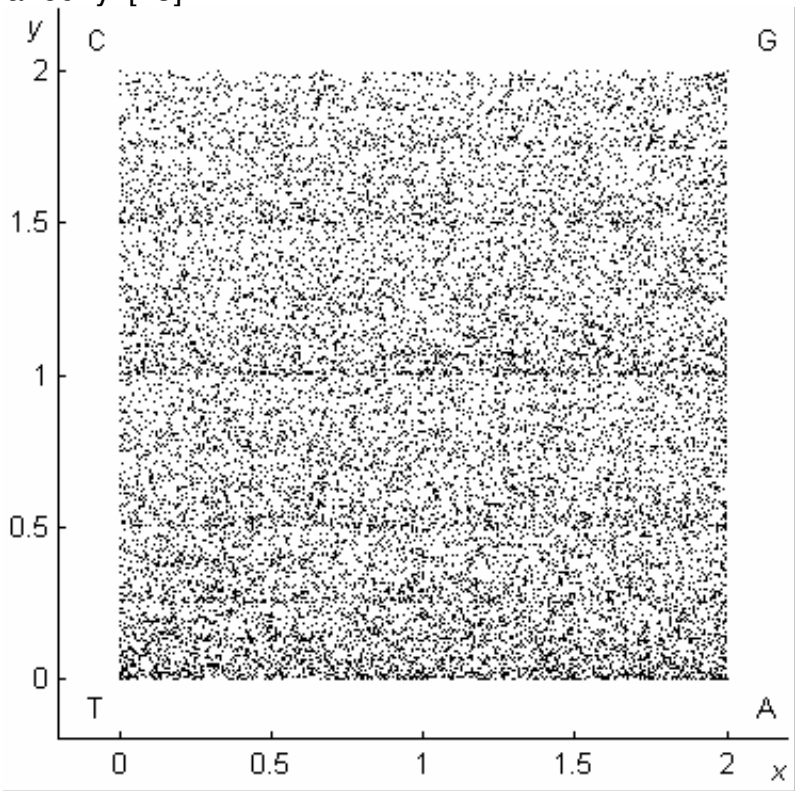
Obrázek 7.1.: Vypočtený obraz HEXA hexosaminidázy A.
[Generováno vlastním programem v Matlabu – *dna2ctver*]

Definován je takto: ‚Homo sapiens hexosaminidase A (alpha polypeptide) (HEXA) on chromosome 15'. Nese označení lokus **NG_009017** s 32 743 bp (base pair = bázových párů). Sekvence je obsažena v lidském geonomu. Procentuální zastoupení bází je následující, A 26,9 %, C 20,4 %, G 25,0 % a T 27,7 %. Výběr této sekvence byl na popud desátého cvičení z předmětu Úvod do biomedicínského inženýrství (BUMI), kde byl autor naveden přímo na ni. [23]

Výsledný obrázek ještě není po žádné úpravě. Navíc je zde nevýhoda, kdy Matlab vypočte body s vysokou přesností, a při ukládání do obrázku se tato přesnost nepřenese – zaokrouhlí se na nejbližší ukládaný pixel.

7.2. Části prvního chromozomu

Celá definice zkoumané části je ‚Homo sapiens chromosome 1, alternace assembly Celera, whole genome shotgun sequence'. Sekvence nese označení: lokus **AC_000044** se 17 479 bp. Tato sekvence je součástí lidského geonomu. Zastoupení bází má podíl 28,2 % pro bázi A, 20,9 % pro bázi C, 22,4 % pro bázi G a 28,5 % pro bázi T. Obrázek vytvořený z této sekvence je níže (**Obrázek 7.2**). Výběr této sekvence byl náhodný. [23]



Obrázek 7.2.: Vypočtený obraz pro úsek prvního chromozomu. [Generováno vlastním programem v Matlabu – *dna2ctver*]

7.3. ATP7A

Definice sekvence zní „Homo sapiens ATPase, Cu++ transporting, alpha polypeptide (ATP7A) on chromosome X", dána lokusem **NG_013224** a délkou 139 699 bp. Sekvence kóduje transmebrální protein, který umožňuje průchod měďnatým iontům skrz buněčnou stěnu. A báze je zastoupena z 28,9 % v sekvenci, C báze z 19,3 %, G báze z 19,4 % a T báze z 32,5 %.

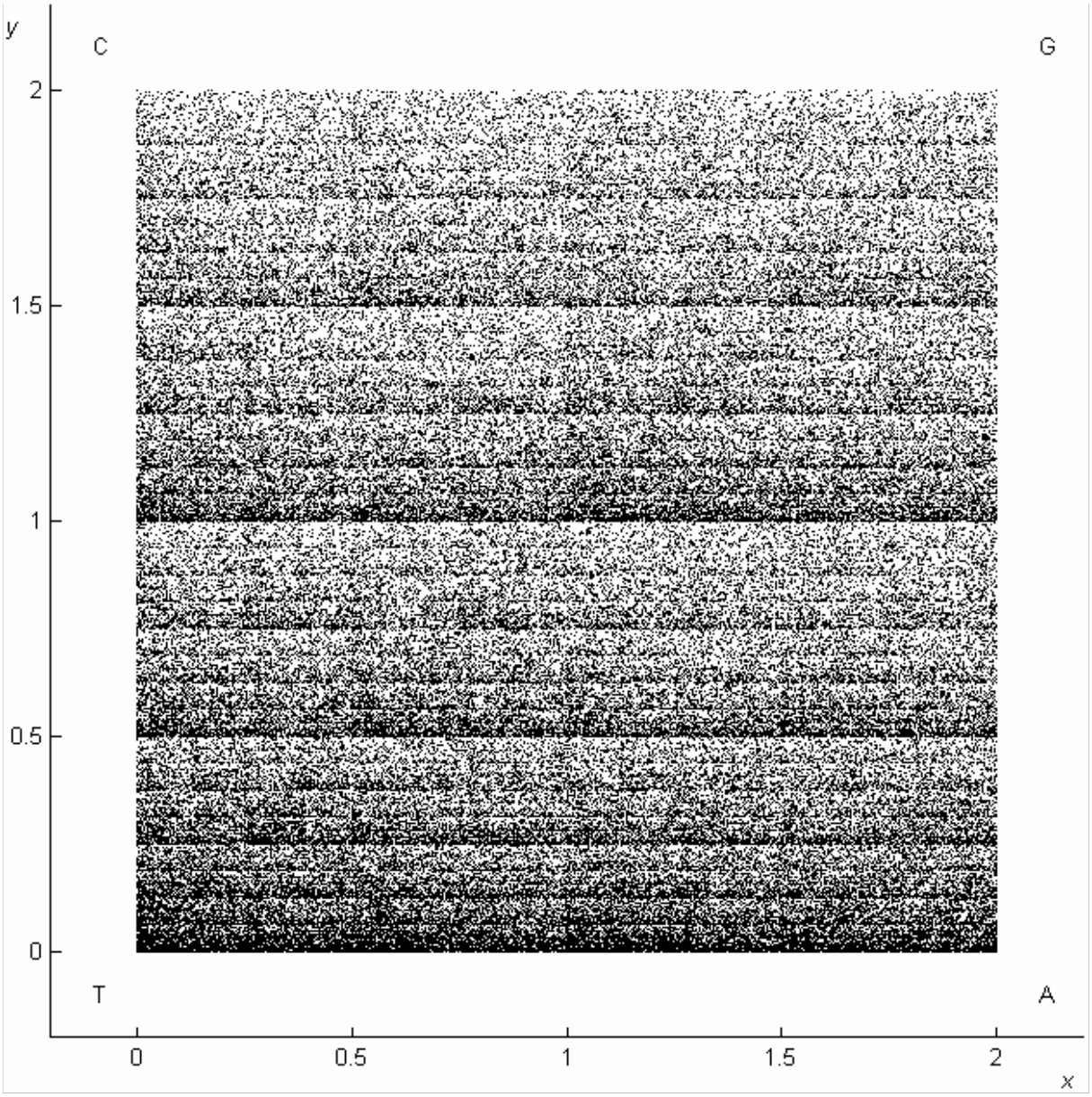
Výstup programu je dále (**Obrázek 7.3**), na kterém je také vidět procentuální zastoupení v sekvenci – jasně převládá dolní polovina obrázku (vyšší zastoupení bází A a T). Výběr této sekvence byl již cílenější – rozhodující parametr byla délka sekvence. Pro následné BCM zpracování byly první dva obrázky příliš řídké a výsledky by neměly žádnou vypovídající hodnotu.

7.4. ATCC 29220

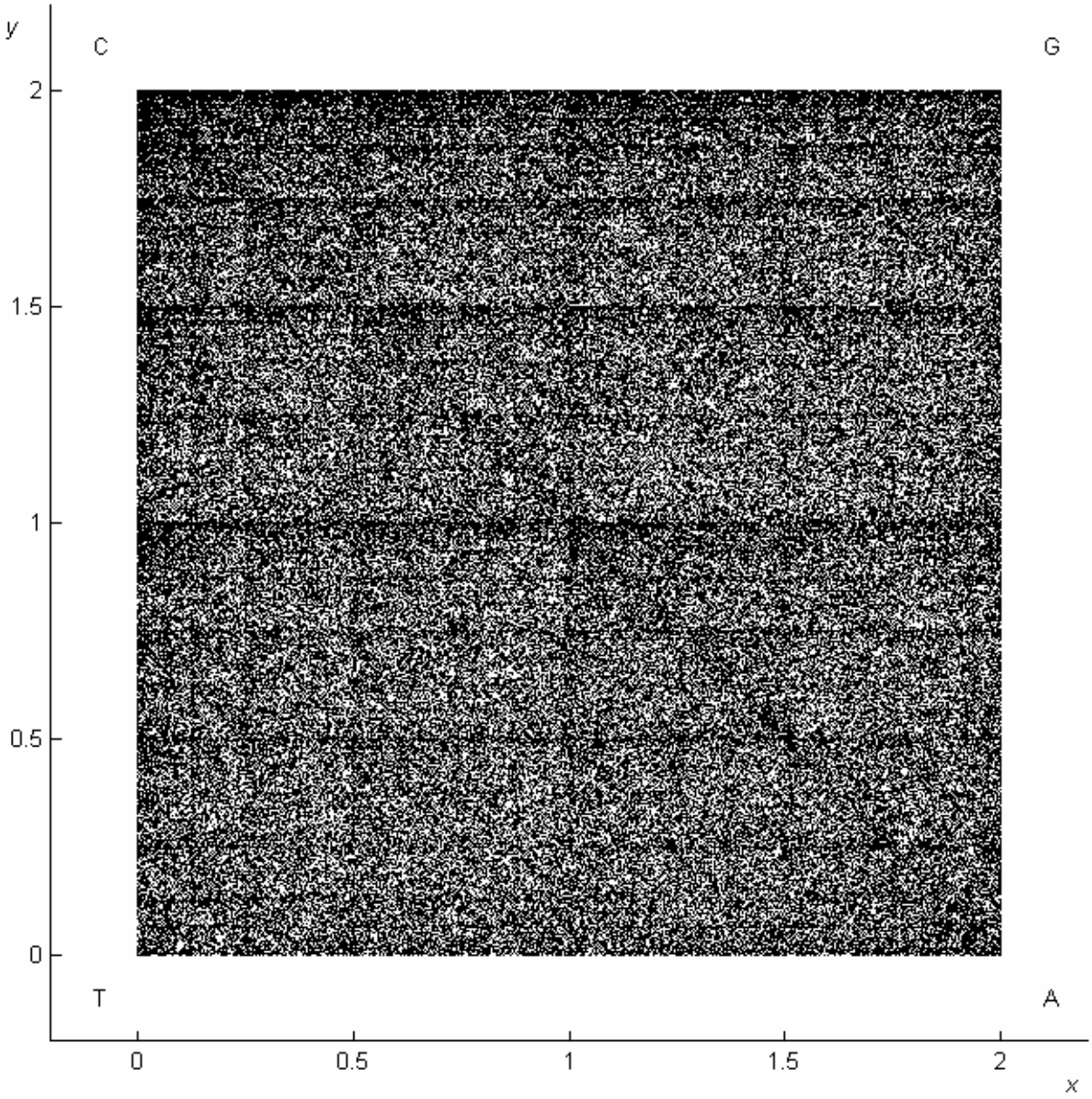
Sekvence je definována: „Citrobacter youngae ATCC 29220 C_sp-1.0.1_Cont0.7, whole genome shotgun sequence", dána lokusem **NZ_ABWL02000007** a délkou 274 831 bp. Sekvence pochází z genetické výbavy bakterie (Citrobacter youngae

ATCC 29220). Sekvence obsahuje z 23,7 % bázi A, z 27,7 % bázi C, z 25,2 % bázi G a z 23,4 % bázi T.

Vykreslený obraz sekvence je dále (Obrázek 7.4), na němž je vidět délka sekvence, kdy obrázek je značně tmavý, a vetší zastoupení báze C. U výběru této sekvence byl opět hlavní parametr její délka.



Obrázek 7.3.: Sekvence NG_013224. [Generováno vlastním programem v Matlabu – *dna2ctver*]



Obrázek 7.4.: Reprezentace sekvence NZ_ABWL02000007. [Generováno vlastním programem v Matlabu – *dna2ctver*]

7.5. Více způsobů vykreslení čtverců ACGT

7.5.1. Vykreslení sekvence po kodónech

Vykreslení bylo poupraveno tak, aby se vždy vykreslila jedna tečka pro trojici znaků jdoucích po sobě. Vrcholy čtverce zůstávají stejně a popis kodónu sedí se segmentací čtverce uvedeného v kapitole 6.2. Segmentování čtverce ACGT, popřípadě je naznačena na obrázku pod ní (**Obrázek 6.2**).

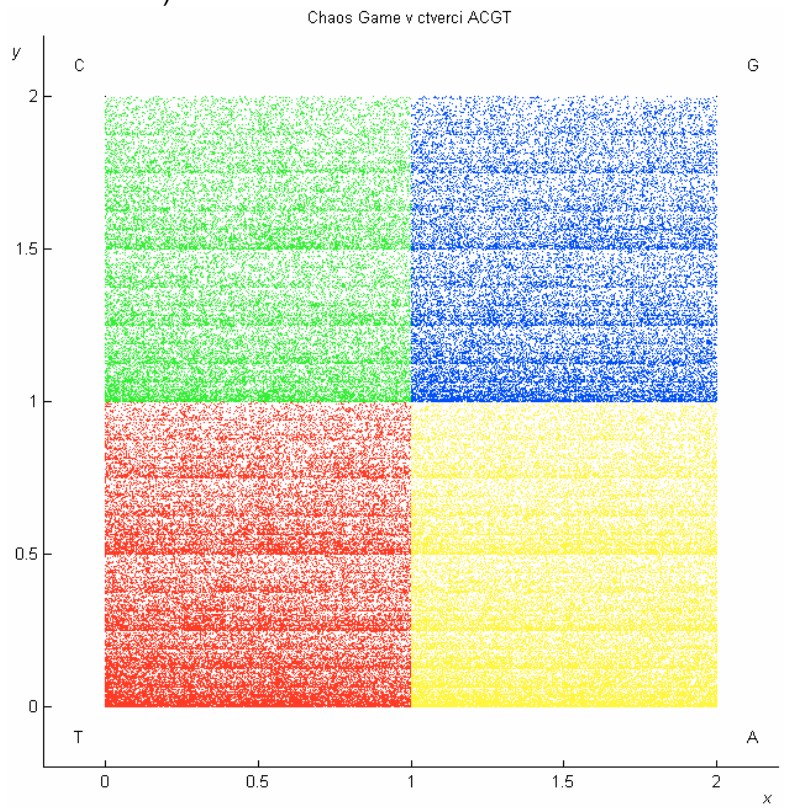
Toto vykreslení zajišťuje program *dna2ctver_kodon*. Vstupní proměnné jsou stejné jako u programu *dna2ctver* a mají také stejné funkce (viz kap. 7.1. HEXA hexosaminidáza A).

7.5.2. Vykreslování sekvence z aminokyselinových sekvencí

Matlab obsahuje funkci k převedení sekvence aminokyselin na kodóny (respektive nukleotidy) a je označena *aa2nt*. Tato funkce je použita ve funkci *dna2ctver_protein* pro převedení aminokyselin na kodóny a posloupnost nukleotidů. Zpracování je pak stejné jako u původního *dna2ctver*. Vstupní proměnné opět stejné jako *dna2ctver* (popsáno v kap. 7.1. HEXA hexosaminidáza A).

7.5.3. Barevné vykreslení bodů

Funkce *dna2ctver* byla doplněna vykreslováním bodů v různých barvách. Funkce nese pojmenování *dna2ctver_barva* a má stejné parametry jako funkce výchozí (*dna2ctver*). Barva je určena bází v posloupnosti DNA, kdy bázi A znázorňuje barva žlutá, C zelená, G červená a T modrá. Výstupní obrázek funkce je níže pro sekvenci NG_013224 (**Obrázek 7.5**).

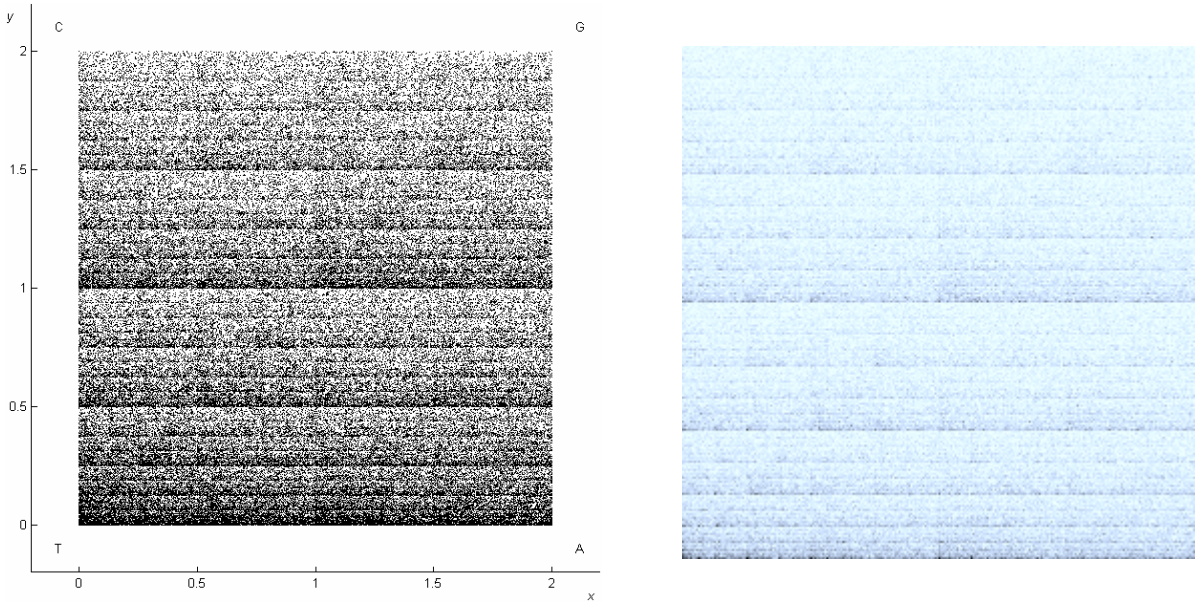


Obrázek 7.5.: Barevné provedení NG_013224. [Generováno vlastním programem v Matlabu – *dna2ctver_barva*]

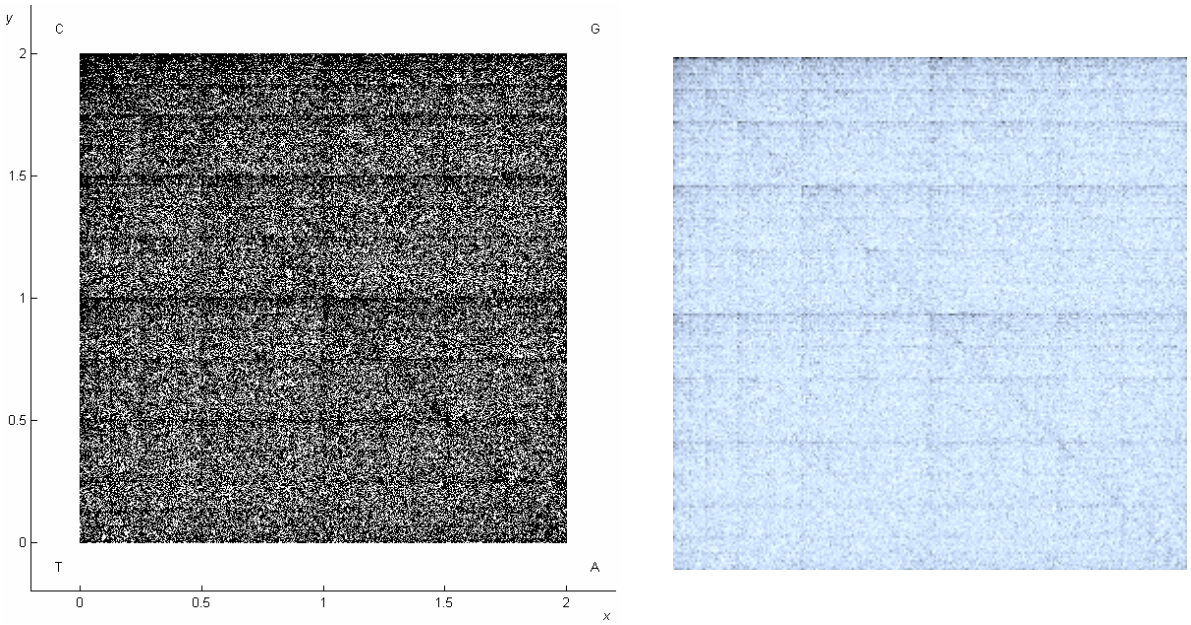
7.5.4. Jiný způsob vykreslení

Myšlenka zrychlení programu přinesla nový pohled na způsob vykreslení DNA do čtverce. Zavrhla se myšlenka vykreslování do souřadných os, nahradila ji čtvercová matice, kde se metodou Chaos Game (viz kap. 6.1. Úprava metody chaos game pro zkoumání sekvence DNA) inkrementují nejbližší vypočtené souřadnice o 1. Po vykreslení celé matice se v matici najde maximum, normuje se, to znamená podělení maximální hodnotou a rozsah je pak od 0 do 1, a celá matice se invertuje – aby nositelkou informace byla barva černá. Výsledný obraz je pak v šedotónové stupnici barev a ne jen v černobílé.

Takto pracující funkce je označena *dna2ctver_matice*, a má několik výhod oproti původní funkci *dna2ctver* (srovnání na sekvenci NG_013224 uvedené v kap. 7.3. ATP7A). Zaprvé je rychlejší. To potvrzují časy výpočtu stejné sekvence, kdy původní program vykreslil obrázek asi za minutu a čtvrt, zatímco s novým přístupem čas výpočtu klesl asi na dvacet sekund. Další výhoda je možnost uživatelem definované velikosti výstupního obrázku, jako hlavní parametr funkce *dna2ctver_matice(velikost)*. U původního programu se musel vykreslený obrázek ručně uložit a ořezat a velikost takto upraveného obrázku byla maximálně 500 x 500 pixelů. S možností volby velikosti si můžeme jednak velikost přímo navolit, ale navíc výsledný obrázek se uloží ve stejné velikosti, jakou jsme zadali, a není třeba jej dále ořezávat. Další výhodou pravděpodobně je i menší chyba při zaokrouhlování na pixely. Poslední výhodou je přímé ukládání obrázku ve formátu bmp. Je zde také nevýhoda – výsledný obraz je šedotónový a je nutno jej před dalším zpracováním prahovat, což vnese další chybu. Tento problém může být chápán i jako výhoda, protože dojde k potlačení šumu. Srovnání výstupů obou funkcí je na obrázku dále (**Obrázek 7.6** a **Obrázek 7.7**).



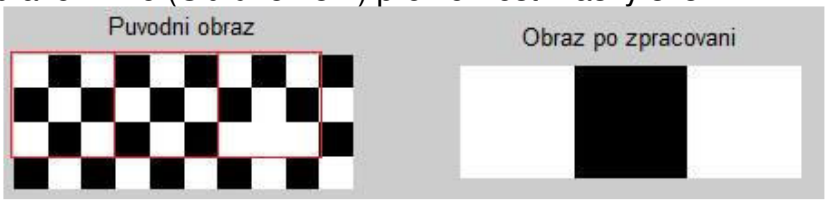
Obrázek 7.6.: Srovnání výstupu programu pro sekvenci NG_013224. Vlevo *dna2ctver* a vpravo *dna2ctver_matice(256)*. [Generováno vlastními programy v Matlabu]



Obrázek 7.7.: Srovnání výstupu programu pro sekvenci NZ_ABWL02000007. Vlevo *dna2ctver* a vpravo *dna2ctver_matice(256)*. [Generováno vlastními programy v Matlabu]

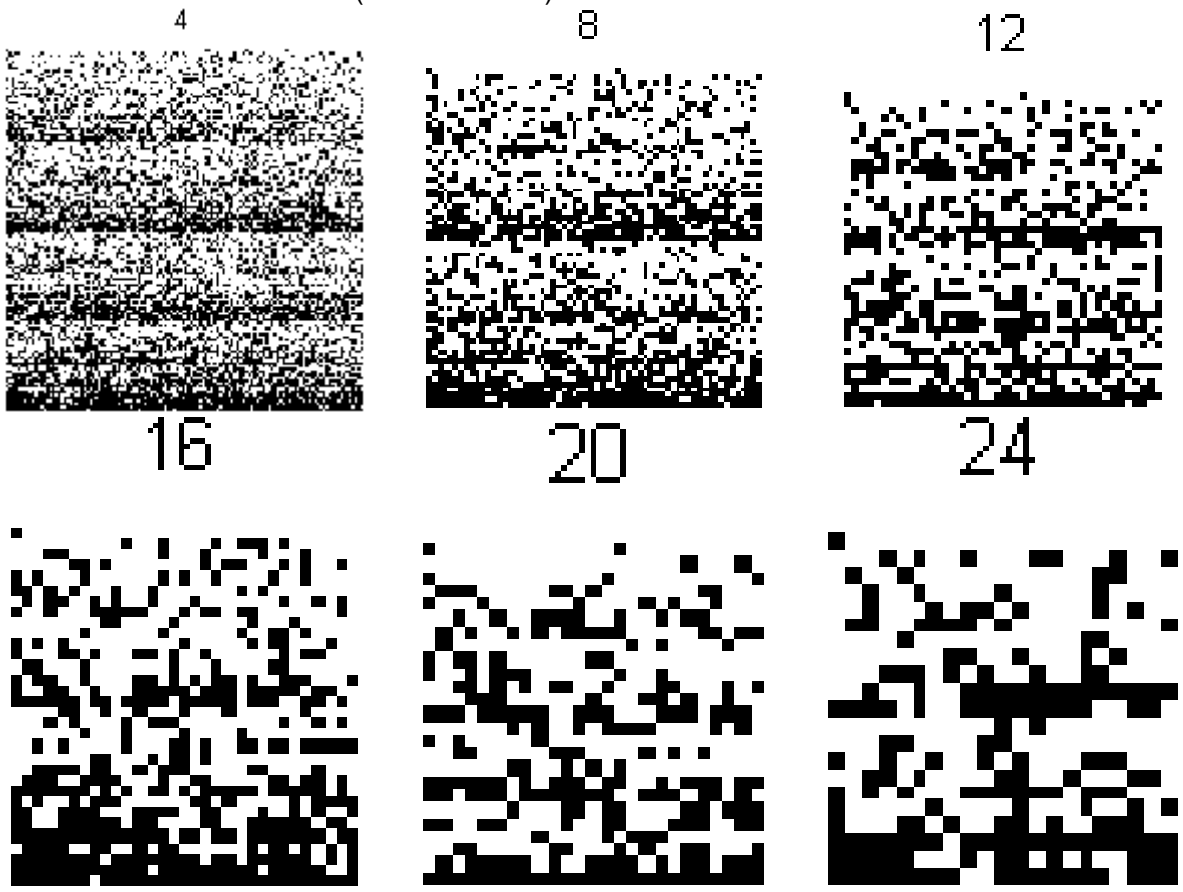
8. ZPRACOVÁNÍ BCM

Zkratka BCM pochází z prvních písmen anglického **B**ox **C**ounting **M**ethod. BCM je zpracování obrazu, které je jednoduché a názorné. V dané metodě jde o „počítání bodů“ nesoucí informaci pod danou maskou. Velikost masky udává dané přiblížení, to znamená, že první přiblížení (velikost masky 1x1) nezmění vstupní obraz, druhé přiblížení (velikost masky 2x2) zmenší původní obraz na polovinu a výsledný obraz je dán poměrem počtu bodů nesoucích informaci v původním obraze k velikosti masky. Takto se stále zvětšuje maska. Pokud je poměr větší než 0,5 je výsledný bod černý, je-li menší bude bod bílý. Vstupní obraz musí být monochromatický. Popsaný postup znázorňuje obrázek níže (**Obrázek 8.1**) pro velikost masky 3x3.



Obrázek 8.1.: Názorná ukázka zpracování obrazu metodou BCM. [Vlastní tvorba]

Obrázek 8.1 naznačuje i problém nedokonalého překrytí původního obrazu a masek. Po rozvaze byl přesah vyloučen z další analýzy, protože by vnášel chybu; doplnění obrázku černými nebo bílými body na celistvý počet masek by vnášel také velkou chybu do dalšího přiblížení. Postupné vykreslení přiblížení je naznačeno na obrázku na obrázku níže (**Obrázek 8.2**).



Obrázek 8.2.: Postupné zpracování BCM. [Generováno vlastním programem v Matlabu - BCMpod]

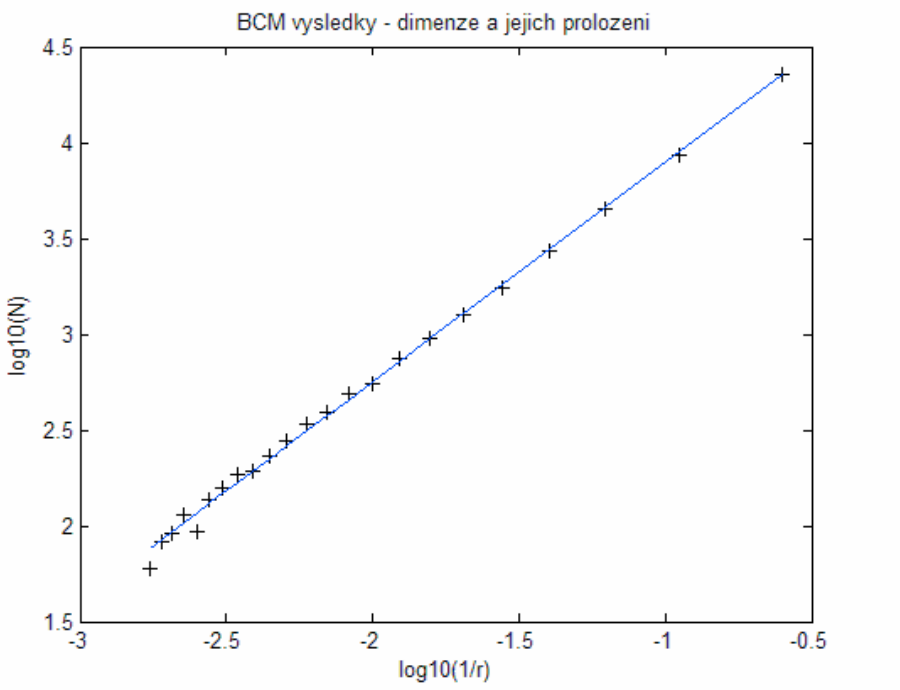
Z obrázků je jasně vidět, že při zpracování se výstupní obrázek zmenšuje – číslo nad obrázky má stále stejnou velikost; pro přehlednost byly malé obrázky zvětšeny. Čísla nad obrázky znamenají krok výpočtu, a také velikost strany masky.

Problém nastává také u sudých velikostí stran masek, tj. sudý počet pixelů masky, kdy může dojít k rovnosti počtu černých pixelů s polovinou plochy masky (poměr 0,5). Pak je problém rozhodnout, zda výstupní bod bude bílý nebo černý. Toto je vyřešeno generováním náhodného (pseudonáhodného) čísla od 0 do 1 a znova se přistoupí k rozhodování. Je to jednoduché řešení, ale opět vnáší chybu do zpracování a reprodukovatelnost výsledků se pak sníží.

V každém přiblížení se vypočte fraktální dimenze podle vzorce (8) v kapitole 1.2.1. Výpočet fraktální dimenze. Ve vzorci *N* znamená počet černých bodů (nesoucí informaci) a *r* počet pixelů pod maskou (maska 3x3 –> 9 pixelů). Spočtené *D* (dimenze) se ukládají a nakonec se vykreslí do grafu, kde osa x představuje *log(1/x)* a osa y *log(N)*. Toto můžeme vidět na obrázku níže (**Obrázek 8.3**). Následné zpracování je naznačeno v následující kapitole (8.1. Zpracování obrázku sekvence NG_013224). Popsaným postupem pracuje program *BCMned(obr,vz)*, kde místo vstupní proměnné *obr* se píše název obrázku pro zpracování v uvozovkách (př.: ‘NG_013224.bmp’) a místo *vz* se píše počet přiblížení, nebo-li velikost maximální velikosti strany masky (default: 24).

8.1. Zpracování obrázku sekvence NG_013224

Zpracování sekvence zpracované programem dna2ctver a zobrazené na obrázku 7.3 (**Obrázek 7.3**) a seznámení se sekvencí proběhlo v kapitole 7.3. ATP7A. Obrázek je ořezán tak, aby osy a okolní bílý podklad nevnášely chybu, a následné zpracování bylo čistě jen pro daný obrázek sekvence. Naznačení kroků zpracování je na obrázku níže (**Obrázek 8.3**).



Obrázek 8.3.: Výstup programu - dimenze a jejich proložení. [Generováno vlastním programem v Matlabu - BCMned]

V obrázku (**Obrázek 8.3**) jsou jednotlivé dimenze – body, proloženy přímkou. Pokud rovnici (8) upravíme do obecného tvaru přímky, která má tvar $y = k \cdot x + q$, dostaneme tento tvar (11)

$$\log(N) = D \cdot \log\left(\frac{1}{r}\right). \tag{11}$$

Člen *q* je v této rovnici zanedbán a pokud rovnici zobecníme pro více dimenzí, pak můžeme v rovnici (11) psát místo *D* konstantu F_k, která popisuje směrnici přímky prokládající dimenze metodou nejmenších čtverců. Rovnice pak vypadá

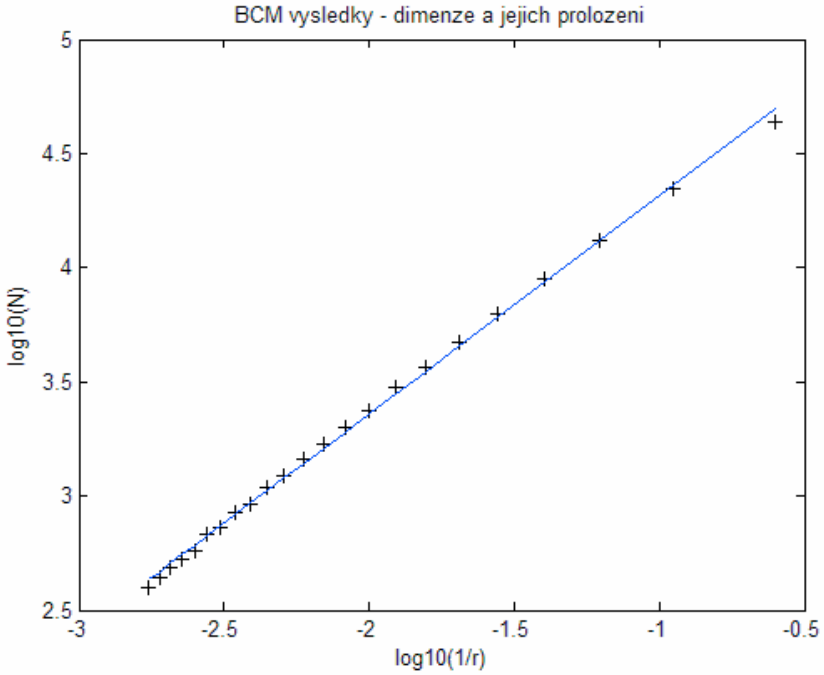
$$\log(N) = F_k \cdot \log\left(\frac{1}{r}\right), \tag{12}$$

kde F_k označíme jako multifraktální koeficient. Touto záměnou byla získána nezávislost na kroku přiblížení (popřípadě dimenzi či velikosti masky) a lze ho pokládat za parametr sekvence DNA pro zpracování metodou BCM.

Výsledný multifraktální koeficient pro sekvenci NG_013224 je $1{,}1445 \pm 0{,}002$. Výsledek je takto udán z důvodů rozptylu výsledných hodnot z důvodu rovnosti počtu černých pixelů vztažených k velikosti masky, které se pak rovnají rozhodovací úrovni (viz kap. 8. Zpracování BCM). Proto byl výpočet asi desetkrát opakován a stanovila se průměrná hodnota a relativní chyba výsledku.

8.2. BCM obrázku sekvence NZ_ABWL02000007

Obrázek vypočtených dimenzí a stanovení multifraktálního koeficientu pro sekvenci NZ_ABWL02000007 je níže (**Obrázek 8.4**).



Obrázek 8.4.: Výstup BCM pro sekvenci NZ_ABWL02000007. [Generováno vlastním programem v Matlabu - BCMned]

Výsledný koeficient sekvence NZ_ABWL0200007 je $0{,}9562 \pm 0{,}0005$.

9. ZPRACOVÁNÍ PSM

Zkratka PSM pochází z anglického názvu Power Spectrum Method a je to metoda zpracování obrazu, kdy se počítá výkonové spektrum obrázku. PSM je výpočetně náročnější oproti BCM. Výpočet je postaven na Fourierově transformaci obrázku (zisk reálné a imaginární složky obrázku, označení $F(k_i)$), která je následně umocněna, a získáme tak výkonové spektrum P_i (situaci popisuje vzorec (13)). [3]

$$P_i = \left|F(k_i)\right|^2 = \left(\sqrt{\mathrm{Re}(F(k_i))^2 + \mathrm{Im}(F(k_i))^2}\right)^2 = \mathrm{Re}(F(k_i))^2 + \mathrm{Im}(F(k_i))^2, \quad (13)$$

kde Re(x) je reálná složka frekvenčního spektra, Im(x) je imaginární složka frekvenčního spektra a k_i je prostorová frekvence na i-té hodnotě. [14]

Dále je nutno vzít v úvahu model ideálního výkonového spektra $\hat{P}_i$, který získáme jako model jednodimenzionálního fraktálního systému. [14]

$$\hat{P}_i = c \cdot \frac{1}{|k_i|^{\beta}} = c \cdot |k_i|^{-\beta}, \quad (14)$$

kde *c* je konstanta výkonového spektra a β je exponent vztahující se k fraktální dimenzi. [14]

Cílem této metody je nalezení právě konstanty β, ze které je pak také vypočtena dimenze. Výpočet konstanty β je realizován podle vzorce (15):

$$\beta = \frac{N\sum_{i=1}^{N}(\ln P_i)(\ln|k_i|) - \left(\sum_{i=1}^{N}\ln|k_i|\right)\left(\sum_{i=1}^{N}\ln P_i\right)}{\left(\sum_{i=1}^{N}\ln|k_i|\right)^2 - N\sum_{i=1}^{N}(\ln|k_i|)^2}, \quad (15)$$

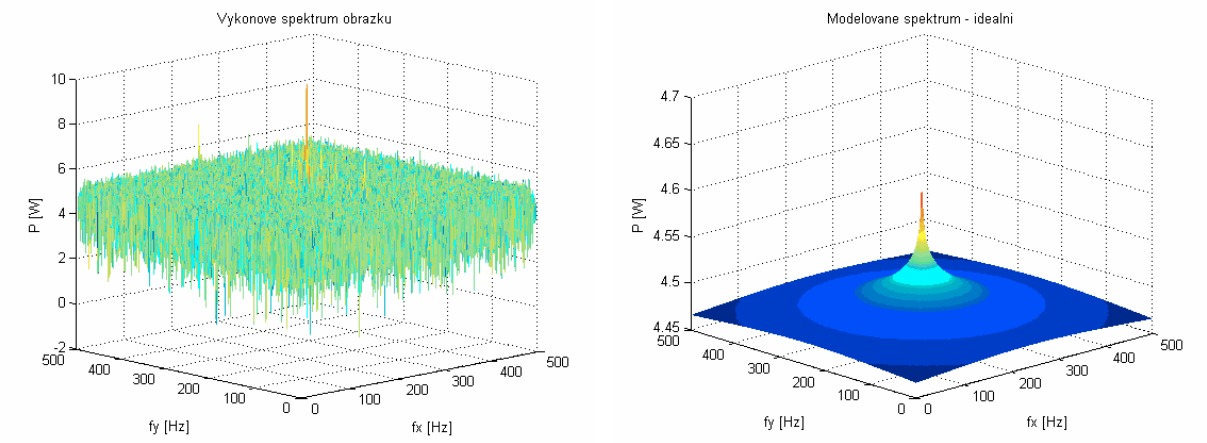
kde *N* udává počet vzorků spektra. Výpočet PSM dimenze už je pak snadný a znázorňuje ji vzorec (16):

$$D_{PSM} = \frac{5-\beta}{2}. \quad (16)$$

Celé podrobné odvození je popsáno ve zdroji [14], a nebo i s příkladem a obrázky jednotlivých kroků výpočtu ve zdroji [3].

9.1. Program *PSMned*

V programu Matlab byl vytvořen program PSMned, který počítá velikost fraktální dimenze výše popsaným postupem. Vstupní proměnná je obrázek, který se má tímto způsobem zpracovat. Příklad zadání programu pro výpočet dimenze je následující: *PSMned('NZ_ABWL02000007_bez_okraju.bmp')*. Program vykreslí tři okna. V prvním okně jsou dvě části, které zobrazují vstupní obrázek a výkonové spektrum P_i. Ve druhém okně je pak prostorově vykresleno výkonové spektrum obrázku a v posledním okně je model ideálního spektra také zobrazen v prostoru (viz **Obrázek 9.1**). Po vykreslení oken se v Command Window (pracovní plocha Matlabu) objeví velikost spočtené dimenze. Pro ukázkový obrázek sekvence NZ_ABWL02000007 vyjde dimenze 2,4763.



Obrázek 9.1.: Výkonové spektrum obrázku sekvence NZ_ABWL02000007 (vlevo, druhé okno) a jeho model (vpravo, třetí okno). [Generováno vlastním programem v Matlabu – *PSMned*]

10. POROVNÁNÍ VÝSLEDKŮ

Správnou činnost programů je potřeba ověřit. Porovnání bude provedeno mezi metodami BCM a PSM, pak BCM s ověřeným programem *frak* Ing. M. Vally (získáno z CD přiloženého u zdroje [3]), a PSM s vhodně nastaveným programem *PSdetekce* (stejný zdroj, [3]).

10.1. Srovnání BCM a PSM

Každá metoda počítá fraktální dimenzi jiným způsobem, a tak se dají předpokládat odlišné výsledky. Výsledky pro čtyři sekvence jsou uvedeny v následující tabulce (**Tabulka 6**). První dvě sekvence nelze brát zcela vážně – mají menší počet přiblížení (méně než 25).

Tabulka 6: Srovnání výsledků BCM a PSM. [Výstupy vlastních funkcí *BCMned* a *PSMned*]

Obrázek sekvence	*BCMned*	*PSMned*
NG_009017	2,0995 *	2,4719
AC_000044	2,7223 **	2,4751
NG_013224	1,1445	2,4850
NZ_ABWL02000007	0,9562	2,4763

kde * - možných pouze 11 přiblížení, ** - možných pouze 5 přiblížení.

10.2. Srovnání BCM

Zde jde o srovnání činnosti programu *BCMned* a *frak*, který byl získán ze zdroje [3]. Program *BCMned* byl upraven tak, aby mohl zpracovávat libovolný obrázek převedený do tvaru vhodného pro zpracování, tj. obrázek byl převeden na černobílý, byly v něm vyzvednuty hrany a nakonec byl invertován. To proto, aby informaci o hranách nesly černé pixely. Následné zpracování obrazu poskytlo multifraktální koeficient, a ten je srovnán s dosaženými čísly ve zdroji [3] pro daný obrázek. Výsledky v tabulce níže (**Tabulka 7**) jsou výsledky po 25 přiblíženích. U posledního z uvedených je struktura nevýrazná s malým množstvím výrazných hran, a právě proto došlo pravděpodobně ke zvětšování zkreslení s rostoucí velikostí masky.

Tabulka 7: Srovnání výsledků BCM. Porovnání výstupu funkce *BCMned* s čísly v [3].

Obrázek	*frak*	*BCMned*
kůra stromu	1,1117	1,1632
seno	1,2246	1,3021
tráva	0,914	0,9316
písek	1,358	1,4633

10.3. Srovnání PSM

Zde bude ověření provedeno opačným způsobem – obrázek sekvence DNA bude vložen do vhodně nastaveného programu *PSdetekce* (viz [3]). Vypočtené hodnoty jsou srovnány v tabulce níže (**Tabulka 8**) pro oba programy.

Tabulka 8: Srovnání PSM. Výpočet dimenze programy *PSdetekce* [3] a *PSMned*.

Obrázek sekvence	*PSdetekce*	*PSMned*
NG_009017	2,472	2,4719
AC_000044	2,475	2,4751
NG_013224	2,485	2,4850
NZ_ABWL02000007	2,476	2,4763

11. ZÁVĚR

Práce obsahuje naprogramované funkce pro vypočet Sierpinského trojúhelníku deterministickým způsobem (*deterSiTr*) a metodou chaos game (*chaosSiTr*) uvedené v kap. 3. Sierpinského trojúhelník, a jejich zdrojové kódy z prostředí Matlab uvedeny v přílohách Příloha 1 a 2. Zdrojový kód metody chaos game byl poté upraven na čtverec ACGT pro zkoumání posloupnosti bází v DNA sekvenci. Program *upravaDNAkodu* (Příloha 3) upravuje sekvence do potřebného tvaru. Program vrací normovanou sekvenci DNA, kterou dále zpracuje program *dna2ctver* (Příloha 4). Výstupem této funkce bude obrázek bodů ve čtverci ACGT, ve kterém bude hledána fraktální souvislost. Příklady výstupu jsou čtverce na obrázcích Obrázek 7.1, Obrázek 7.2. Další modifikací programu je možnost barevného vykreslení *dna2ctver_barva* (viz Obrázek 7.5), nebo *dna2ctver_protein*, který umožní vykreslení proteinových sekvencí do DNA čtverce, dále je to program *dna2ctver_kodon*, který vykresluje sekvence po kodónech, tj. pro tři po sobě jdoucí znaky vykreslí jednu tečku. Jako zvláštní modifikace je změna způsobu vykreslení sekvence programem *dna2ctver_matice* (Příloha 5), které je srovnáno na obrázcích Obrázek 7.6 a Obrázek 7.7. Hlavní výhoda oproti programu *dna2ctver* je jeho výpočetní rychlost.

Dále je zde zpracování získaných obrázků sekvencí. Metoda BCM je použita ve vytvořeném programu *BCMned* (Příloha 6), který vrací hodnotu multifraktálního koeficientu. Hodnota multifraktálního koeficientu je nezávislá na kroku přiblížení, a tak se stává parametrem této sekvence. Takto by bylo možné klasifikovat sekvence DNA. Druhý způsob zpracování je metoda zkratkou nazvaná PSM. Sestavený program *PSMned* (Příloha 7) pracuje touto metodou a počítá velikost fraktální dimenze. Všechny stěžejní programy vyjmenované výše jsou v Přílohách 1 – 7 na konci práce. Všechny programy jsou na přiloženém CD. Součástí práce je srovnání výsledků (viz kap. 10. Porovnání výsledků). Je zde srovnání BCM a PSM (kap. 10.1. Srovnání BCM a PSM). Výsledky se dosti liší, což je zapříčiněno pravděpodobně odlišným způsobem výpočtu. Pak srovnání jednotlivých metod výpočtu fraktální dimenze s externími zdroji (kap. 10.2. Srovnání BCM a 10.3. Srovnání PSM). Výsledky BCM jsou odlišné, což je přičteno stavu, kdy je přesně polovina pixelů masky zaplněna černými pixely, a pak rozhodnutí o černém či bílém bodu je ponecháno na náhodně vygenerovaném čísle (proto i rozptyl výsledků je větší). PSM vychází naprosto stejně jako v programu ze zdroje [3] (viz Tabulka 8).

Tato bakalářská práce ukazuje možnosti vzájemného srovnávání různých sekvencí, kdy lze porovnávat již vykreslené čtverce ACGT mezi sebou, popřípadě jejich výkonová spektra, či číselně pomocí multifraktálního koeficientu (výsledek BCM) nebo fraktální dimenze (výsledek PSM). Podle těchto čísel lze i dané sekvence klasifikovat, protože každá sekvence má vlastní velikost multifraktálního koeficientu i fraktální dimenze. I samotné optické (pohledové) srovnání sekvencí může napovědět něco o jejich struktuře, příklad – pokud bude v sekvenci výrazně převyšovat počet bází A nad všemi ostatními, pak bude čtverec ACGT nerovnoměrně zaplněn a to tak, že jeho pravá dolní čtvrtina bude hustěji poseta body; nebo bude-li v sekvenci nejčastější „přechod“ bází (posloupnost dvou znaků za sebou) A-T nebo T-A, bude ve čtverci ACGT nejvýraznější spodní linie čtverce, apod.

Bakalářská práce zdaleka nepopisuje všechny metody zpracování. Dá se přidat ještě hodně dalších vylepšení fraktální analýzy, například filtrací 2D obrazu BCM mohou vzniknout různé vzory charakteristické opět pro danou sekvenci (viz zdroj [28]). Programy se dají ještě také vylepšit, například načítáním sekvencí přímo

z internetové databáze (například NCBI), nebo je spojit v jeden program a přidat GUI s mnoha interaktivními volbami, a třeba i doplněna statistickými metodami pro zpracování DNA sekvencí. Také lze úplně změnit způsob vykreslování. Lze vykreslit sekvenci i do podoby tetrahedtonu, tj. do 3D struktury (pro více informací viz kap. 5.4. 3-D nanostruktura DNA nebo více viz zdroj [26]).

Část této práce, respektive vykreslení ACGT čtverce a jeho zpracování metodou BCM, bylo prezentováno ve studentské soutěži EEICT 2010 jako příspěvek Fractal in Images of DNA Sequence Data.

Seznam použité literatury:

- **Elektronické zdroje:**

[2] HINNER, M.: Jemný úvod do fraktálů. 1993. Dostupné z WWW: http://martin.hinner.info/math/Fraktaly/

[4] SIXTA, T.: Chaos, fraktály, atraktory. Dělení fraktálů. Dostupné z WWW: http://chaos.fraktaly.sweb.cz/strs/3/deleni.html

[5] VANČURA, J.: Fraktály. Kapitola – Fraktální geometrie. Dostupné z WWW: http://www.fractals.webz.cz/fraktalygeo.htm

[6] MARŠÁK, J.: Fraktály a jejich popis v Matlabu. 2003. Dostupné z WWW: http://goro.byl.cz/fraktal_cz.html

[7] VANČURA, J.: Fraktály. Kapitola – Příklady. Dostupné z WWW: http://www.fractals.webz.cz/fraktalypri.htm

[8] WEISSTEIN, E. W.: Lorenz Attractor. From MathWorld – A Wolfram Web Ressource. Dostupné z WWW: http://mathworld.wolfram.com/LorenzAttractor.html

[9] HOŘČICA, A.: Adam's Notepad^2 – Náhodný fraktál. Dostupné z WWW: http://notepad.jslab.net/tvorba/fraktaly/nahodny-fraktal.html

[10] Exploring the Deep Frontier – DNA History. Dostupné z WWW: http://www.ceoe.udel.edu/extreme2004/genomics/dnahistory.html

[11] Molecular Station – DNA Structure. Dostupné z WWW: http://www.molecularstation.com/images/DNA-structure.gif

[12] Sparknotes: Structure of Nucleic Acid; Bases, Sugars and Phosphates. Dostupné z WWW: http://www.sparknotes.com/biology/molecular/structureofnucleicacids/section2.rhtml

[13] SIXTA, T.: Chaos, fraktály, atraktory. Pojmy. Dostupné z WWW: http://chaos.fraktaly.sweb.cz/strs/1/pojmy.html

[16] Sparknotes: Structure of Nucleic Acid; Nucleotides and Nucleic Acids. Dostupné z WWW: http://www.sparknotes.com/biology/molecular/structureofnucleicacids/section1.rhtml

[17] Volně přístupný polský e-learning: GENETYKA. Dostupné z WWW: http://e-learning5.webpark.pl/genetics.htm

[18] LEJA, D.: National Human Genome Research Institute. Dostupné z WWW: http://www.accessexcellence.org/RC/VL/GG/dna_replication.php

[19] J. Craig Venter Institute: What's a Genome? Dostupné z WWW:
http://www.genomenewsnetwork.org/resources/whats_a_genome/Chp1_1_1.shtml

[20] PEDZERA, J.: 3-D struktura lidského geonomu má formu fraktálu. Dostupné z WWW:
http://www.osel.cz/index.php?clanek=4677

[21] TIŠNOVSKÝ, P.: Fraktály v počítačové grafice. Dostupné z WWW:
http://www.root.cz/clanky/fraktaly-v-pocitacove-grafice-i/
http://www.root.cz/clanky/fraktaly-v-pocitacove-grafice-ii/
http://www.root.cz/clanky/fraktaly-v-pocitacove-grafice-xxx/

[23] National Center for Biotechnology Information: HEXA hexosaminidase A (sloha polypeptide). Dostupné z WWW:
http://www.ncbi.nlm.nih.gov/sites/entrez?db=gene&cmd=retrieve&dopt=full_report&list_uids=3073

- **Klasické zdroje:**

[1] BERTHELSEN, Ch. L.: Fractal analysis of DNA sequence data. The University of Utah, 1993. 160 s.

[3] VALLA, M.: Fraktály v obrazech – Bakalářská práce. Brno: VUT Brno, 2006. 50 s.

[14] TURNER, M. J., BLACKLEDGE, J. M., ANDREWS, P. R.: Fractal Geometry in Digital Imaging. Leicester, Academic Press 1998, ISBN 0-12-703970-8

[15] JELÍNEK, J., TICHÁČEK, V.: BIOLOGIE pro střední školy gymnazijního typu. Olomouc: Fin Publishing, 1996. 415 s., 1. vydání.

[22] KARBONE, A., SEEMAN, N. C.: A route to fractal DNA-assembly. Netherlands: Kluwer Academic Publisher, 2002. 469-480 s.

[24] VALLA, M.: Úvod do biomedicínckého inženýrství: Přednášky, Fraktály. 2009.

[25] KARBONE, A., SEEMAN, N. C., and other: 3D fractal assembly from coding, geometry and protection. Netherlands: Kluwer Academic Publisher, 2004. 235-252 s.

[26] KE, Y., SHARMA, J., LIU, M., JAHN, K., LIU, Y., YAN, H.: Scaffolded DNA Origami of a DNA Tetrahedron Molecular Container. Arizona: American Chemical Society, 2009.

[27] ROTHEMUND, P. W. K., PAPADAKIS, N., WINFREE, E.: Algorithmic Self-Assembly of DNA Sierpinski Triangles. PloS Biol 2(12): e424, 2004.

[28] ASHLOCK, D., GOLDEN, J. B. III: Iterated Function System Fractal for the Detection and Display of DNA Reading Frame. Iowa, State University, 2009.

Seznam zkratek a příloh:

Seznam zkratek:

kap.	=	kapitola
obr.	=	obrázek
tab.	=	tabulka
tj.	=	to je, to jsou
apod.	=	a podobně

Seznam příloh:

Příloha 1: Zdrojový kód z prostředí Matlab pro konstrukci Sierpinského trojúhelníka deterministickým způsobem. Program *deterSiTr*.

Příloha 2: Zdrojový kód z prostředí Matlab pro konstrukci Sierpinského trojúhelníka metodou Chaos game. Program *chaosSiTr*.

Příloha 3: Zdrojový kód z prostředí Matlab pro úpravu získaného kódu z veřejné databáze do formy vhodné pro zpracování. Program *upravaDNAkodu*.

Příloha 4: Zdrojový kód z prostředí Matlab pro vykreslení DNA čtverce pro danou posloupnost bází. Program *dna2ctver*.

Příloha 5: Zdrojový kód z prostředí Matlab pro vykreslení DNA čtverce pomocí matice, kde se počítá četnost dopadu běhu programu, ta se pak normuje a vykreslí jako obrázek. Program *dna2ctver_matice*.

Příloha 6: Zdrojový kód z prostředí Matlab pro výpočet multifraktálního koeficientu metodou BCM. Program *BCMned*.

Příloha 7: Zdrojový kód z prostředí Matlab pro výpočet fraktální dimenze metodou PSM. Program *PSMned*.

Obsah přiloženého CD:

- Elektronická verze bakalářské práce ve formátu PDF
- m-file všech vytvořených programů (pro prostředí Matlab)
- txt s pojmenováním lokusu sekvence obsahující uvnitř
- výstupní (popřípadě vstupní) obrázky programů

Přílohy

Příloha 1: Zdrojový kód z prostředí Matlab pro konstrukci Sierpinského trojúhelníka deterministickým způsobem. Program *deterSiTr*.

```
function deterSiTr(N)
%Funkce deterSiTr(N) vykresli N-tou iteraci Sierpinskeho trojuhelniku
%deterministickym zpusobem. N udava pocet iteraci; default: N=3

A=[0 0]; B=[2 0]; C=[1 1.732051];   %def. vrcholu trojuhelnika
plot(A(1),A(2),'ko','MarkerSize',3,'MarkerFaceColor','k');
                                                %vykresleni bodu A, B a C
hold on;
text(-0.1,-0.1,'A');
plot(B(1),B(2),'ko','MarkerSize',3,'MarkerFaceColor','k');
text(2.1,-0.1,'B');
plot(C(1),C(2),'ko','MarkerSize',3,'MarkerFaceColor','k');
text(0.975,1.83,'C');
text(2.15,-0.3,'\itx'); text(-0.3,1.95,'\ity');     %popis os
axis square; axis([-0.2 2.2 -0.2 2]); box off;      %uprava zobrazeni
plot(line([A(1) B(1)],[A(2) B(2)],'Color','k'));
                                                %vykresleni stran trouhelnika
plot(line([B(1) C(1)],[B(2) C(2)],'Color','k'));
plot(line([A(1) C(1)],[A(2) C(2)],'Color','k'));
patch([A(1) B(1) C(1)],[A(2) B(2) C(2)],'k');       %vybarveni na cerno

if (nargin == 0)            %defaultni nastaveni
    N=3;
end

N=N+1;        %pocet iteraci +1
%max 9 (8 iteraci) - vypocet do 2 minut
%pro 10 (9 iteraci) - vypocet 5 minut
mv=zeros((3^N-1)/2+3,6);
mv(1,:)=[A(1) A(2) B(1) B(2) C(1) C(2)];

if N==1             %podminka pro spravne vykresleni nulte iterace
    hold off;
else
    for i=1:((3^N-1)/2)
        ax=(mv(i,3)+mv(i,5))/2;     %vypocet vrcholu
        ay=(mv(i,4)+mv(i,6))/2;
        bx=(mv(i,1)+mv(i,5))/2;
        by=(mv(i,2)+mv(i,6))/2;
        cx=(mv(i,1)+mv(i,3))/2;
        cy=(mv(i,2)+mv(i,4))/2;
        mv(2+(i-1)*3,:)=[mv(i,1) mv(i,2) bx by cx cy];
                                    %ulozeni vrcholu do matice
        mv(3+(i-1)*3,:)=[ax ay mv(i,3) mv(i,4) cx cy];
        mv(4+(i-1)*3,:)=[ax ay bx by mv(i,5) mv(i,6)];
        plot(line([ax bx],[ay by],'Color','w'));
                                    %vykresleni stran novych trojuhelniku
        plot(line([ax cx],[ay cy],'Color','w'));
        plot(line([bx cx],[by cy],'Color','w'));
        patch([ax bx cx],[ay by cy],'w');       %vybarveni na bilo
    end
end
```

Příloha 2: Zdrojový kód z prostředí Matlab pro konstrukci Sierpinského trojúhelníka metodou Chaos game. Program *chaosSiTr*.

```
function chaosSiTr( N, mark, prod, a, b )
%Funkce chaosSiTr(N,mark,prod,a) vykresli N iteraci Sierpinskeho
%trojuhelniku metodou
%chaos game.
%N udava pocet vykreslenych bodu; default: N=10000.
%mark udava velikost vykreslovanych bodu; default: mark=1.
%prod udava velikost zpozdeni pri vykreslovani; default: prod=0.
%a udava moznost zapnuti (a=1) vykresleni stran trojuhelniku;
%default: a=0 (vypnuto).
%b udava moznost zapnuti (b=1) vykresleni spojnic bodu a zvoleneho vrcholu;
%default: b=0 (vypnuto).

close all;
A=[0 0]; B=[2 0]; C=[1 1.732051];  %def. vrcholu trojuhelnika
plot(A(1),A(2),'ko','MarkerSize',3,'MarkerFaceColor','k');
                                    %vykresleni bodu A, B a C
hold on;
text(-0.1,-0.1,'A');
plot(B(1),B(2),'ko','MarkerSize',3,'MarkerFaceColor','k');
text(2.1,-0.1,'B');
plot(C(1),C(2),'ko','MarkerSize',3,'MarkerFaceColor','k');
text(0.975,1.83,'C'); title('Sierpinskeho trojuhelnik');
text(2.15,-0.3,'\itx'); text(-0.3,1.95,'\ity');     %popis os
axis square; axis([-0.2 2.2 -0.2 2]); box off;      %uprava zobrazeni
if (nargin == 0)                     %definice chybejicich parametru
    N=10000; mark=1; prod=0.0001; a=0; b=0;
end
if (nargin == 1)
    mark=1; prod=0.0001; a=0; b=0;
end
if (nargin == 2)
    prod=0.0001; a=0; b=0;
end
if (nargin == 3)
    a=0; b=0;
end
if (nargin == 4)
    b=0;
end

if a==1                 %vykresleni stran trouhelnika
    plot(line([A(1) B(1)],[A(2) B(2)],'Color','k'));
    plot(line([B(1) C(1)],[B(2) C(2)],'Color','k'));
    plot(line([A(1) C(1)],[A(2) C(2)],'Color','k'));
end

P=[1 0.866];            %pocatecni bod uprostred trojuhelnika
for i=0:N
     r=mod(round(10*rand),3)+1;  %pseudonahodna volba 1, 2 a 3
     switch r
         case 1
             if b==1
                plot(line([A(1) P(1)],[A(2) P(2)],'Color','g'));
                pause(prod/2);
                plot(line([A(1) P(1)],[A(2) P(2)],'Color','w'));
             end
             P(1)=(P(1)+A(1))/2; P(2)=(P(2)+A(2))/2;
```

```
            plot(P(1),P(2),'ko','MarkerSize',mark,'MarkerFaceColor','k');
            pause(prod);
        case 2
            if b==1
               plot(line([B(1) P(1)],[B(2) P(2)],'Color','g'));
               pause(prod/2);
               plot(line([B(1) P(1)],[B(2) P(2)],'Color','w'));
            end
            P(1)=(P(1)+B(1))/2; P(2)=(P(2)+B(2))/2;
            plot(P(1),P(2),'ko','MarkerSize',mark,'MarkerFaceColor','k');
            pause(prod);
        case 3
            if b==1
               plot(line([C(1) P(1)],[C(2) P(2)],'Color','g'));
               pause(prod/2);
               plot(line([C(1) P(1)],[C(2) P(2)],'Color','w'));
            end
            P(1)=(P(1)+C(1))/2; P(2)=(P(2)+C(2))/2;
            plot(P(1),P(2),'ko','MarkerSize',mark,'MarkerFaceColor','k');
            pause(prod);
    end
end
```

Příloha 3: Zdrojový kód z prostředí Matlab pro úpravu získaného kódu z veřejné databáze do formy vhodné pro zpracování. Program *upravaDNAkodu*.

```
function dna = upravaDNAkodu(seq)
%Upravi DNA kod - odstrani mezery a jine znaky nez ACGT
%priklad zapisu: upravaDNAkodu(seq)
%nacteni z txt: seq=importdata('DNAHEXA.txt')

seq=char(seq);      %zmena sekvece na char
seq=seq(:);         %zmena matice na vektor
seq=lower(seq);     %zmena vsech pismen na male
i=1;
for s=1:length(seq)
        switch seq(s)
            case 'a'
                dna(i)=seq(s);
                i=i+1;
            case 'c'
                dna(i)=seq(s);
                i=i+1;
            case 'g'
                dna(i)=seq(s);
                i=i+1;
            case 't'
                dna(i)=seq(s);
                i=i+1;
        end
end
```

Příloha 4: Zdrojový kód z prostředí Matlab pro vykreslení DNA čtverce pro danou posloupnost bází. Program *dna2ctver*.

```
function dna2ctver(mark, prod)
%Funkce dna2ctver(mark, prod) prevede usek DNA kodu ulozeneho v souboru
%txt na obraz bodu ve čtverci ACGT.
%mark = vyjadruje velikost zobrazovanych bodu
%  = prirozene cislo (doporuceno 1,2,3); default: mark=1.
%prod=casove zpozdeni vykreslovani (prodleva); default: prod=0.

A=[0 0]; C=[2 0]; T=[2 2]; G=[0 2];       %def. vrcholu ctverce
plot(A(1),A(2),'ko','MarkerSize',3,'MarkerFaceColor','k');
hold on;                                  %vykresleni bodu A, C, T a G
text(-0.1,-0.1,'A');
plot(C(1),C(2),'ko','MarkerSize',3,'MarkerFaceColor','k');
text(2.1,-0.1,'C');
plot(T(1),T(2),'ko','MarkerSize',3,'MarkerFaceColor','k');
text(2.1,2.1,'T');
plot(G(1),G(2),'ko','MarkerSize',3,'MarkerFaceColor','k');
text(-0.1,2.1,'G'); title('Chaos Game v ctverci ACGT');

text(2.15,-0.3,'\itx'); text(-0.3,2.15,'\ity');     %popis os
axis square; axis([-0.2 2.2 -0.2 2.2]); box off;    %uprava zobrazeni

plot(line([A(1) C(1)],[A(2) C(2)],'Color','k'));  %vykresleni stran ctverce
plot(line([C(1) T(1)],[C(2) T(2)],'Color','k'));
plot(line([T(1) G(1)],[T(2) G(2)],'Color','k'));
plot(line([G(1) A(1)],[G(2) A(2)],'Color','k'));

if(nargin == 0)              %defaultni nastaveni
    mark=1; prod=0;
end

seq=importdata('DNAHEXA.txt');      %ziskani dat z txt souboru
retezec=upravaDNAkodu(seq);         %upraveni DNA kodu do potrebne formy

P=[1 1];        %pocatecni bod uprostred ctverce
plot(P(1),P(2),'ro','MarkerSize',mark,'MarkerFaceColor','r');
for i=1:length(retezec)
     switch retezec(i)        %srovnani pismene na i-te pozici v retezci
         case 'a'
             P(1)=(P(1)+A(1))/2; P(2)=(P(2)+A(2))/2;
             plot(P(1),P(2),'ko','MarkerSize',mark,'MarkerFaceColor','k');
             pause(prod);
         case 'c'
             P(1)=(P(1)+C(1))/2; P(2)=(P(2)+C(2))/2;
             plot(P(1),P(2),'ko','MarkerSize',mark,'MarkerFaceColor','k');
             pause(prod);
         case 'g'
             P(1)=(P(1)+T(1))/2; P(2)=(P(2)+T(2))/2;
             plot(P(1),P(2),'ko','MarkerSize',mark,'MarkerFaceColor','k');
             pause(prod);
         case 't'
             P(1)=(P(1)+G(1))/2; P(2)=(P(2)+G(2))/2;
             plot(P(1),P(2),'ko','MarkerSize',mark,'MarkerFaceColor','k');
             pause(prod);
     end
end
```

Příloha 5: Zdrojový kód z prostředí Matlab pro vykreslení DNA čtverce pomocí matice, kde se počítá četnost dopadu běhu programu, ta se pak normuje a vykreslí jako obrázek. Program *dna2ctver_matice*.

```
function dna2ctver_matice(velikost)
%Funkce dna2ctver_matice(velikost) vykresli sekvenci DNA do ctverce metodou
%Chaos game. Vystupem funkce je ctvercovy obrazek o velikosti parametru
%presnost.
%
%Priklad zadani:    dna2ctver_matice        (Velikost obrazku 512 x 512)
%                   dna2ctver_matice(128)   (Velikost obrazku 128 x 128)
%

if(nargin == 0)             %defaultni nastaveni
    x=512;
else
    x=velikost;
end

matice=zeros(x,x);
C=[0 0]; T=[x 0]; A=[x x]; G=[0 x]; P=[ceil(x/2) ceil(x/2)];

seq=importdata('NG_013224.txt');  %DNAHEXA.txt, prvnichromozom.txt,
NG_013224.txt, NZ_ABWL02000007.txt
retezec=upravaDNAkodu(seq);
%retezec='aaaa';    %pro odladeni chyb-misto dvou predchozich radku

for i=1:length(retezec)
     switch retezec(i)
         case 'a'
             P=round((P+A)/2);
             matice(P(1),P(2))=matice(P(1),P(2))+1;
         case 'g'
             P=round((P+G)/2);
             matice(P(1),P(2))=matice(P(1),P(2))+1;
         case 'c'
             P=round((P+C)/2);
             matice(P(1),P(2))=matice(P(1),P(2))+1;
         case 't'
             P=round((P+T)/2);
             matice(P(1),P(2))=matice(P(1),P(2))+1;
     end
end

matice=(1-(matice/max(max(matice))));   %prevedeni na normované hodnoty (0-
1) a invertovani barev - cerna, jako nosital informace
imshow(matice);     %zobrazeni sekvence
imwrite(matice,'Obr1.bmp');    %ulozeni sekvence
```

Příloha 6: Zdrojový kód z prostředí Matlab pro výpočet multifraktálního koeficientu metodou BCM. Program *BCMned*.

```
function BCMned(obr,vz)
%Funkce BCMned vraci vypoctenou hodnotu fraktalniho koeficientu pro zvoleny
%pocet priblizeni (vz). Pocita cerne body!
%
%Vzor zapisu:   BCMned(obr,vz)
%               BCMned('NG_013224_bez_okraju.bmp',24)
%   obr - cesta ke zkoumanemu obrazku
%   vz - pocet priblizeni (min. 2; max. 24)

if(nargin == 0)     %definice chybejicich vstupnich promennych
    obr='NG_013224_bez_okraju.bmp'; vz=24;
elseif(nargin == 1)
    vz=24;
end

dim=zeros(5,vz);    %matice dimenzi a prislusnych cisel k jejich vypoctu
dim(1,:)=1:vz;

c=imread(obr);      %nacteni obr
[m,n]=size(c);      %zjisteni velikosti obrazku
dim(5,1)=m;
dim(6,1)=n;
figure(1); subplot(461); imshow(c); title('Original');     %vykresleni

for b=2:vz
    poc=0;              %promenna pro pocitani cernych pixelu
    d=ones(floor(m/b),floor(n/b));        %definice bileho obr
    for x=1:b:m         %rozkouskovani obrazu na velikost masky - b x b
        for y=1:b:n
            if (x+b-1)<=m      %osetreni presahu masky pres rozmer obrazku
                if (y+b-1)<=n
                    mpoc=0;
                    for k=x:x+b-1     %prochazeni casti obr pod maskou
                        for l=y:y+b-1
                            if c(k,l)==0
                                mpoc=mpoc+1;    %pocitani cernych bodu pod
maskou v puvodnim obr
                            end
                        end
                    end
                    if (mpoc/b^2)>0.5
                        d(ceil(x/b),ceil(y/b))=0;   %definice cernych pozic
noveho obr
                        poc=poc+1;      %pocitani cernych bodu v novem obr
                    end
                    if (mpoc/b^2)==0.5
                        z=rand();       %osetreni rovnosti
                        if z<0.5
                            d(ceil(x/b),ceil(y/b))=0;
                            poc=poc+1;
                        end
                    end
                end
            end
        end
    end
    %d=d(1:b:end,1:b:end);  %patricne zmenseni noveho obr
```

```
    [a1,a2]=size(d);
    subplot(4,6,b); imshow(d); title(b);    %vykresleni
    %figure(b); imshow(d); title(b);
    dim(2,b)=poc;   %pocet cernych bodu
    dim(3,b)=b^2;   %pocet pixelu pod maskou
    dim(4,b)=log10(poc)/log10(1/b^2);   %dimenze obrazu na danem priblizeni
    dim(5,b)=a1;    %vyska obrazku - pro kontrolu
    dim(6,b)=a2;    %sirka obrazku - pro kontrolu
end

disp('Krok'); disp('Velikost masky v pixelech'); disp('Dimenze');
disp('Vyska obrazku'); disp('Sirka obrazku');
dim([1,3,4,5,6],:)

figure(2);          %vykresleni dimenzi pro vsechna spoctena priblizeni
osaX=log10(1./dim(3,2:vz));
osaY=log10(dim(2,2:vz));
plot(osaX,osaY,'k+');
hold on;              %vykresleni dimenzi a prolozeni
smer=POLYFIT(osaX,osaY,1);
proloz = smer(1)*osaX + smer(2);
plot(osaX,proloz);      %prolozeni bodu (dimenzi)
xlabel('log10(1/r)'); ylabel('log10(N)');
title('BCM vysledky - dimenze a jejich prolozeni');
hold off;
disp('Fraktalni koeficient'); smer(1)
```

Příloha 7: Zdrojový kód z prostředí Matlab pro výpočet fraktální dimenze metodou PSM. Program *PSMned*.

```
function PSMned(obr)
%Funkce PSMned(obr) vraci vypoctenou dimenzi obrazku ziskanou metodou
%vykonoveho spektra.
%
%Vzor zapisu:   PSMned
%                   -prednastavena sekvence
%               PSMned('NG_013224_bez_okraju.bmp')
%                   -sekvence NG_013224

if (nargin == 0)
    obr = 'NZ_ABWL02000007_bez_okraju.bmp';
end

a=imread(obr);      %nacteni
a=double(a);
subplot(121); imshow(a); title('Original');     %vzkresleni puvodniho obr

b=fft2(a);          %DFT (FFT)
b=fftshift(abs(b));
[m,n]=size(b);      %zjisteni velikosti obr

p=real(b).^2+imag(b).^2;    %vykonove spektrum

% priprava pro vypocet vzorce - vypocet dimenze
n1d=m*n;    %pocet vzorku spektra
Xs=linspace(-(n/2),n/2,n);
```

```
Ys=linspace(-(m/2),m/2,m);
[X,Y]=meshgrid(Xs/(n/2),Ys/(m/2));    %def. prost. vektoru
k=sqrt(X.^2+Y.^2);  %moduly souradnic prost. frekv.
ind=find(k==0);     %nalezeni nuly pro log
k(ind)=1;       %zmena 0 na 1
lnP=log(p);  %vypocet logaritmu p
lnK=log(k);     %vypocet logaritmu k

% vzorec
cita=n1d*sum(sum(lnP.*lnK)) - sum(sum(lnK))*sum(sum(lnP));
jmen=(sum(sum(lnK))^2 - n1d*sum(sum(lnK.^2)));
beta = (cita) / (jmen);     %vypocet bety
C=(1/n1d)*sum(sum(lnP))+(beta/n1d)*sum(sum(lnK));
c=exp(C);           %stanoveni konstanty c
q = c*(k.^-beta);
disp('Dimenze obrazku');
DF=(5-beta)/2      %stanoveni dimenze a jeji vypsani

subplot(122); imshow(log10(p),[]); title('Pi - vykonove spektrum');
figure; mesh(log10(p)); title('Vykonove spektrum obrazku');
figure; mesh(log10(q)); title('Modelovane spektrum - idealni');
```