Univerzita Palackého v Olomouci
Přírodovědecká fakulta
Katedra experimentální fyziky

DIPLOMOVÁ PRÁCE

Model difúze pomocí blokových celulárních automatů

Vypracoval:	**David Smrčka**
Studijní program:	Fyzika
Studijní obor:	Aplikovaná fyzika
Forma studia:	Prezenční
Vedoucí diplomové práce:	Mgr. Jan Říha, Ph.D.
Termín odevzdání práce:	květen 2013

Prohlášení

Prohlašuji, že jsem předloženou diplomovou práci vypracoval samostatně pod vedením ..... a že jsem použil zdrojů, které cituji a uvádím v seznamu použitých pramenů.

V Olomouci dne

.....

Bibliografická identifikace

Jméno a příjmení autora	David Smrčka
Název práce	Model difúze pomocí blokových celulárních automatů
Typ práce	Diplomová
Pracoviště	Katedra experimentální fyziky
Vedoucí práce	Mgr. Jan Říha, Ph.D.
Rok obhajoby práce	2013
Abstrakt	Provedli jsme analýzu pravidel pro blokové celulární automaty a hledali jsme pravidlo, jehož evoluční průběh by se svou charakteristikou co nejvíce blížil difúznímu průběhu. Vybrané pravidlo jsme porovnali s teoretickým popisem difúze, abychom prokázali jeho věrohodnost.
Klíčová slova	Celulární automaty, blokové celulární automaty, Margoliovo sousedství, modelování difúze
Počet stran	82
Počet příloh	1 CD-ROM
Jazyk	český

Bibliographical identification

Autor's first name and surname	David Smrčka
Title	Simulation of diffusion using block cellular automata
Type of thesis	Master
Department	Department of Experimental Physics
Supervisor	Mgr. Jan Říha, PhD
The year of presentation	2013
Abstract	We performed an analysis of the rules for block cellular automata and we were looking for the rule whose evolutionary progress with its characteristic would be closest to diffusion. Selected rule was compared with the theoretical description of diffusion so we could prove its credibility.
Keywords	Cellular automata, block cellular automata, Margolius neighborhoood, simulation of diffusion
Number of pages	82
Number of appendices	1 CD-ROM
Language	czech

Obsah

Úvod

Kdykoliv dojde ke styku dvou a více látek stejného skupenství, dochází mezi nimi nevyhnutelně k difúzi (dochází k jejich promísení). Je to jeden z transportních jevů, které hýbou světem kolem nás. Proto je velmi důležité tento neustále probíhající proces dokázat popsat [1, 2, 3, 4]. Když dokážeme definovat jeho průběh, jsme schopni provádět jeho simulace a právě simulování difúzního procesu jsme si dali za úkol v této diplomové práci.

Konkrétně jsme se zabývali tím, že jsme hledali nejvhodnější pravidlo či pravidla pro blokové celulární automaty, což je specifický typ celulárních automatů, které by se dalo použít právě pro simulování difúze. Podrobně jsme analyzovali všechna vybraná pravidla a vybrali jsme z nich ta nejvhodnější, která jsme poté porovnali s teoretickým popisem difúze, abychom dokázali, že průběh těchto konkrétních pravidel splňuje princip difúzního průběhu.

Blokové celulární automaty nejsou tolik využívány jako ostatní celulární automaty, ale jejich specifické vlastnosti nabízejí dobré využití právě v případech, kdy nám jde o pozorování interakcí mezi částicemi. Proto jsme se rozhodli dokázat, že jsou využitelné kupříkladu právě pro simulování difúzního průběhu.

Průběhy všech celulárních automatů, jejich analýzy a výsledná porovnávání s teorií jsme realizovali v prostředí softwaru *Mathematica*.

Na závěr bychom ještě rádi zmínili, že myšlenka zabývat se simulací difúze pomocí celulárních automatů, přesněji pomocí blokových celulárních automatů, vznikla na třítýdenní letní škole Wolfram Science Summer School 2012, která se uskutečnila v Bostonu, MA, USA, a které se autor diplomové práce zúčastnil.

1. Teorie difúze

Jak již bylo zmíněno dříve, difúze je jedním z transportních jevů, při kterém dochází k samovolnému pronikání částic jedné látky mezi částice druhé látky téhož skupenství, čímž nám ze dvou čistých látek vznikne látka, která je jejich směsí.

Skotský chemik Thomas Graham byl první, kdo experimentálně studoval difúzi v letech 1831–1833. Při studiu vzájemné interakce dvou plynů o různých hustotách pozoroval, že se plyny neuspořádávají tak, že by se lehčí plyn přesunul nad těžší, místo toho docházelo k jejich vzájemnému nahodilému promísení [5].

V roce 1855 německý fyzik Adolf Eugene Fick definoval zákony, pomocí kterých lze popsat průběh difúze. Fick vycházel ze zákona pro tok tepla, který Jean Baptiste Joseph Fourier definoval v roce 1822, jelikož předpokládal spojení mezi tepelným tokem a difúzi, protože oba jsou ovlivněni náhodným pohybem molekul. Matematická definice difúze, jak ji stanovil Fick, stanovuje, že velikost difúzního toku difundující látky přes jednotkovou plochu je proporcionální ke gradientu koncentrace [6]:

$$J = -D\frac{\partial C}{\partial x}, \tag{1.1}$$

kde J je velikost difúzního toku, D je difúzní koeficient a C je koncentrace difundující látky. Rovnice (1.1) bývá často nazývána Fickův I. zákon. Záporné znaménko představuje skutečnost, že látka difunduje z míst o větší koncentraci do míst s koncentrací nižší. Rychlost difúze ovlivňuje difúzní koeficient, který je pro každou látku specifický. Jeho měřením jsme schopni odlišit látky od sebe [7].

Fickův II. zákon je diferenciální rovnice, kterou lze odvodit z rovnice (1.1), podrobný popis odvození lze najít v Crankovi [6]. Ve výsledku dostaneme rovnici

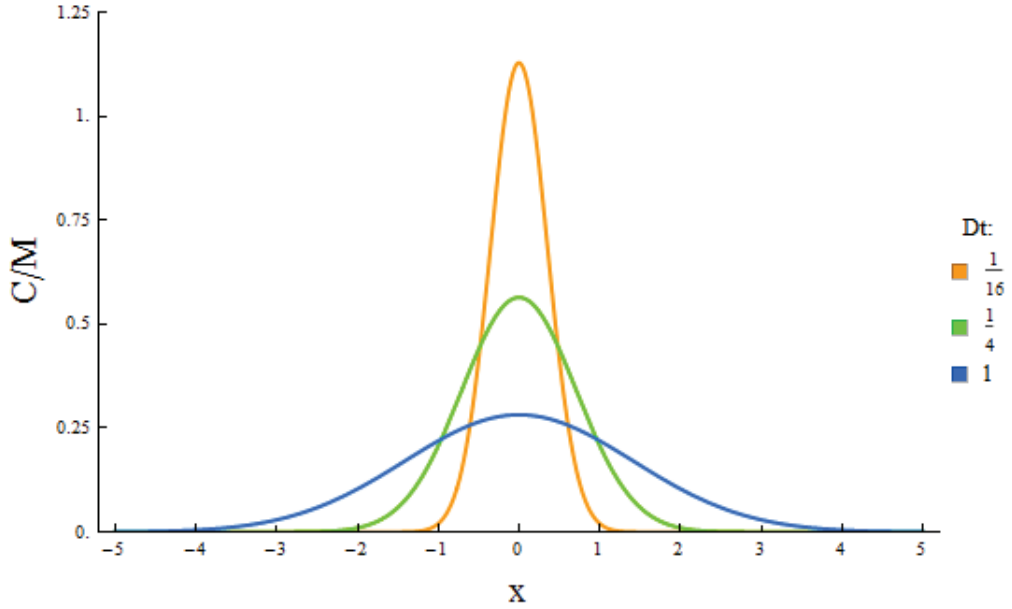
$$\frac{\partial C}{\partial t} = -D^2\frac{\partial^2 C}{\partial x^2}, \tag{1.2}$$

která popisuje změnu koncentrace C v čase t.

Její diferenciací a následnou úpravou dostáváme rovnici

$$C = \frac{M}{2(\pi Dt)^{\frac{1}{2}}} \exp(-x^2/4Dt), \tag{1.3}$$

která popisuje difúzní rozptýlení látky o množství M vložené v čase $t = 0$ do roviny $x = 0$. Takto definovaná rovnice však pro nás nemá příliš valného užitku, jelikož jsme



Obrázek 1.1: Křivky koncentrace v jednotlivých časových úsecích

pracovali s předpokladem, že námi simulovaná látka není umístěna v rovině, nýbrž se nachází na konkrétní ploše konečných rozměrů. Z toho důvodu jsme hledali řešení rovnice (1.3) pro dva různé případy umístění látky: buď je počáteční distribuce ohraničená z jedné strany, nebo je počáteční distribuce ohraničená z obou stran.

1.1 Počáteční distribuce ohraničená z jedné strany

V prvním případě jsou počáteční podmínky definovány následovně:

$$C = C_0, \quad x < 0, \quad C = 0, \quad x > 0, \quad t = 0. \tag{1.4}$$

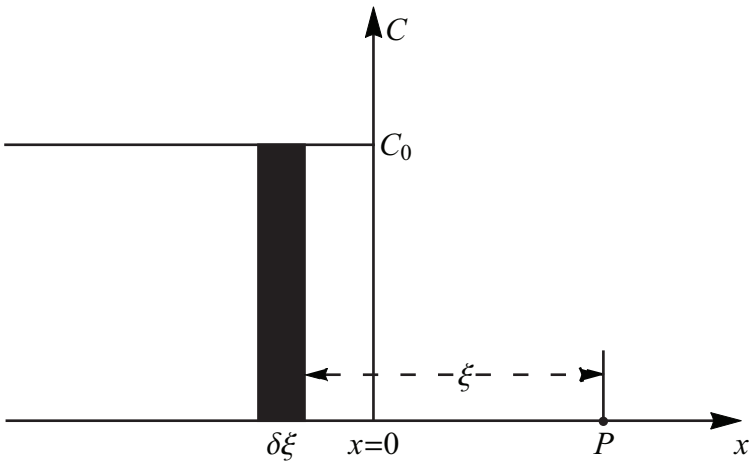
Takto definované počáteční podmínky se dají přirovnat kupříkladu k válci čisté vody, který je postaven na válec s roztokem. Předpokládejme, že difundující látka, která je umístěná v prvku $\delta\xi$ je čárový zdroj o síle $C_0\,\delta\xi$, potom podle rovnice (1.3) je koncentrace v bodě P, ve vzdálenosti ξ od prvku, a v čase t rovna

$$\frac{C_0\,\delta\xi}{2(\pi Dt)^{\frac{1}{2}}} \exp(-\frac{\xi^2}{4Dt}), \tag{1.5}$$

a konečné řešení, dané počáteční distribucí (1.4), je získáno sumací přes všechny následující prvky dξ, tedy

$$C(x,t) = \frac{C_0}{2(\pi Dt)^{\frac{1}{2}}} \int_x^\infty \exp(-\xi^2/4Dt)\,\mathrm{d}\xi = \frac{C_0}{\pi^{\frac{1}{2}}} \int_{x/2\sqrt{Dt}}^\infty \exp(-\eta^2)\mathrm{d}\eta, \tag{1.6}$$

kde $\eta = \xi/2\sqrt{Dt}$.



Obrázek 1.2: Schéma počáteční distribuce [6]

Existuje známá a velmi podrobně popsaná matematická funkce, kterou zde můžeme použít, a to je chybová funkce, která se značí jako erf z. Nejvíce se používá ve statistice, odkud také plyne její název. Její definice je následovná:

$$\mathrm{erf}\, z = \frac{2}{\pi^{\frac{1}{2}}} \int_0^z \exp(-\eta^2)\mathrm{d}\eta. \tag{1.7}$$

Vlastnosti této funkce jsou takové, že

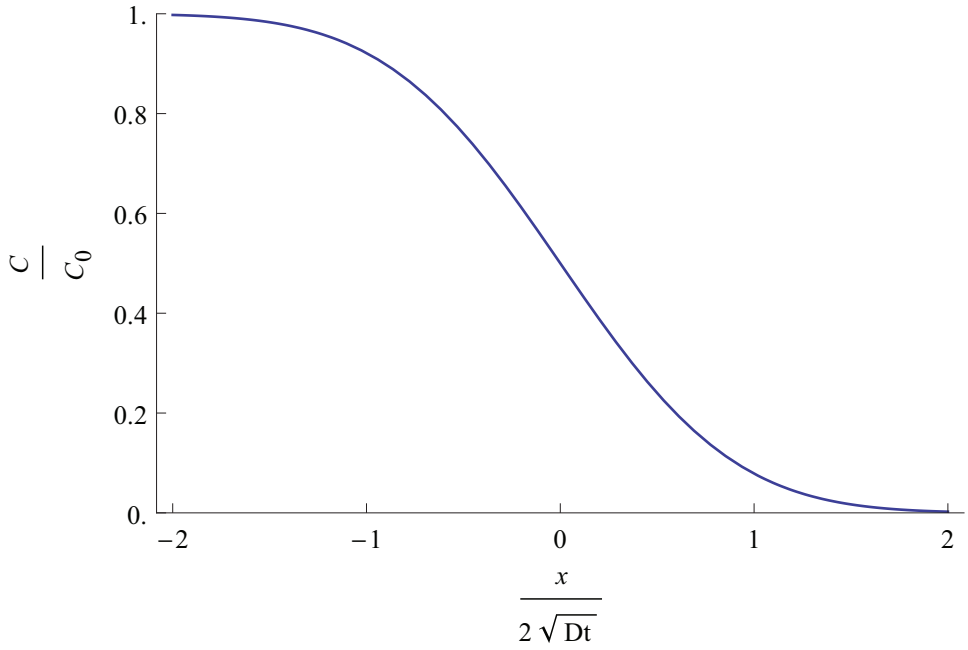
$$\mathrm{erf}\,(-z) = -\mathrm{erf}z, \quad \mathrm{erf}(0) = 0, \quad \mathrm{erf}(\infty) = 1, \tag{1.8}$$

a tudíž, když

$$\frac{2}{\pi^{\frac{1}{2}}} \int_z^\infty \exp(-\eta^2)\mathrm{d}\eta = \frac{2}{\pi^{\frac{1}{2}}} \int_0^\infty \exp(-\eta^2)\mathrm{d}\eta - \frac{2}{\pi^{\frac{1}{2}}} \int_0^z \exp(-\eta^2)\mathrm{d}\eta = 1 - \mathrm{erf}z = \mathrm{erfc}z, \tag{1.9}$$

kde erfc z je komplementární chybová funkce, řešení difúzní rovnice (1.6) můžeme zapsat ve tvaru

$$C(x,t) = \frac{1}{2} C_0 \mathrm{erfc} \frac{x}{2\sqrt{Dt}}. \tag{1.10}$$



Obrázek 1.3: Křivka koncentrace pro systém s jednou stranou konečných rozměrů

Tento případ lze také řešit pomocí Laplaceovy transformace [6]. Je to velmi často používaná integrální transformace, která umožňuje řešit obyčejné diferenciální rovnice tak, že vymizí závislost na čase.

Mějme funkci $f(t)$, kde t musí nabývat kladných hodnot. Potom Laplaceova transformace $\overline{f}(p)$ této funkce je dána jako

$$\overline{f}(p) = \int_0^\infty \exp(-pt) f(t) \mathrm{d}t. \tag{1.11}$$

Pro naše experimentální uspořádání, kdy máme počáteční distribuci ohraničenou z jedné strany, hledáme řešení rovnice pro Fickův II. zákon, viz (1.2), které by splňovalo následující hraniční podmínky

$$C = C_0, \quad x = 0, \quad t > 0, \tag{1.12}$$

a zároveň tyto počáteční podmínky

$$C = 0, \quad x > 0, \quad , t = 0. \tag{1.13}$$

Vynásobením obou stran (1.2) $\exp(-pt)$ a jejich integrací přes t v rozmezí od 0 do ∞ dostaneme

$$\int_0^\infty \exp^{-pt} \frac{\partial^2 C}{\partial x^2} \mathrm{d}t - \frac{1}{D} \int_0^\infty \exp^{-pt} \frac{\partial C}{\partial t} \mathrm{d}t = 0. \tag{1.14}$$

Pokud předpokládáme, že lze vyměnit pořadí mezi diferenciací a integrací, což u funkcí, se kterými pracujeme lze, potom

$$\int_0^\infty \exp^{-pt} \frac{\partial^2 C}{\partial x^2} \mathrm{d}t = \frac{\partial^2}{\partial x^2} \int_0^\infty C \exp^{-pt} \, \mathrm{d}t = \frac{\partial^2 \overline{C}}{\partial x^2}. \tag{1.15}$$

Integrací druhé části (1.14) dostaneme

$$\int_0^\infty \exp^{-pt} \frac{\partial C}{\partial t} \mathrm{d}t = \left[C \exp(-pt) \right]_0^\infty + p \int_0^\infty C \exp(-pt) \mathrm{d}t = p\overline{C}, \tag{1.16}$$

kde výraz v hranatých závorkách vymizí při $t = 0$ v důsledku počátečních podmínek (1.13) a při $t = \infty$ naopak vymizí kvůli exponenciálnímu faktoru. Poté tedy dojdeme k tomu, že se diferenciální rovnice (1.2) zredukuje na tvar

$$D \frac{\partial^2 \overline{C}}{\partial x^2} = p\overline{C}. \tag{1.17}$$

Nyní podobným způsobem vyřešíme hraniční podmínky (1.12), kdy

$$\overline{C} = \int_0^\infty C_0 \exp^{-pt} \, \mathrm{d}t = \frac{C_0}{p}, \quad x = 0. \tag{1.18}$$

Ve výsledku vidíme, že Laplaceova transformace zredukovala parciální diferenciální rovnici (1.2) na běžnou diferenciální rovnici (1.17). Řešením této rovnice, které by splňovalo námi zadané hraniční podmínky, a pro které by $\overline{C}$ zůstávalo konečné zatímco by se x blížilo k nekonečnu je

$$\overline{C} = \frac{C_0}{p} \exp^{-qx}, \tag{1.19}$$

kde $q^2 = p/D$. V tabulkách pro Laplaceovu transformaci lze dohledat, že funkce, jejíž transformace je dána (1.19) má tvar

$$C = C_0 \mathrm{erfc} \frac{x}{2\sqrt{Dt}}, \tag{1.20}$$

kde opět

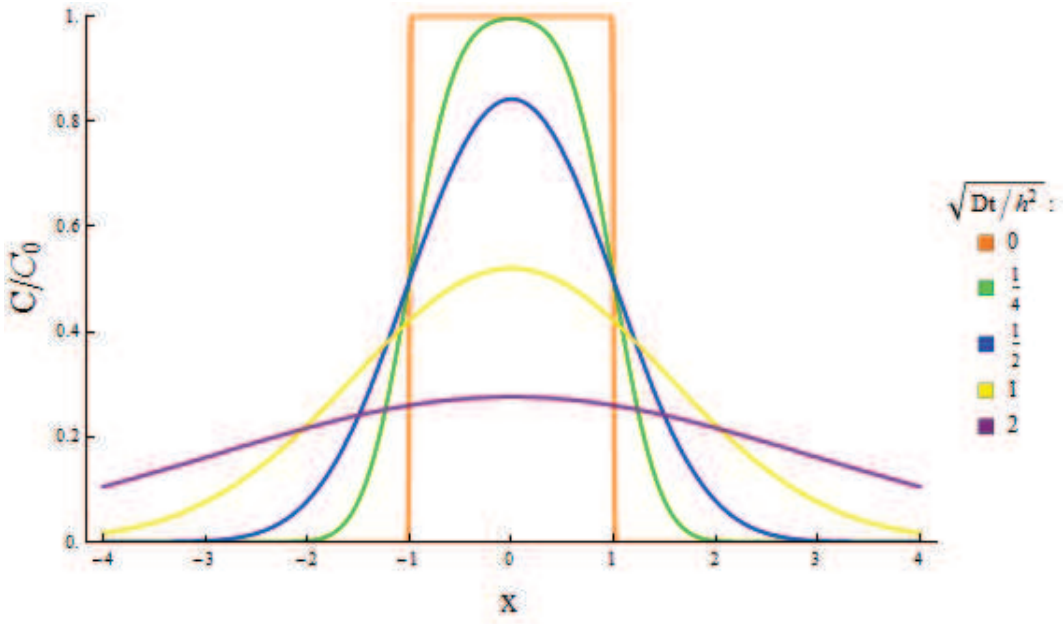
$$\mathrm{erfc}z = 1 - \mathrm{erf}z. \tag{1.21}$$

1.2 Počáteční distribuce ohraničená z obou stran

Podobným způsobem lze přistupovat k řešení (1.3) pro případ, kdy byla látka v čase $t = 0$ ohraničená z obou stran, tedy zaujímala plochu $-h < x < h$, jak je tomu

v Obr. 1.4. V tomto případě integrujeme od $x - h$ po $x + h$ na místo od x do ∞, jak tomu bylo v (1.6). Ve výsledku dostaneme výraz

$$C = \frac{1}{2} C_0 \left\{ \mathrm{erf} \frac{h - x}{2\sqrt{Dt}} + \mathrm{erf} \frac{h + x}{2\sqrt{Dt}} \right\}. \tag{1.22}$$



Obrázek 1.4: Křivka koncentrace pro počáteční distribuci v ohraničeném prostoru

Dosavadní výpočty pracovaly se skutečností, že látka je sice na začátku obsažená v ohraničené oblasti, ale jakmile ji necháme difundovat, bude tak dělat až do nekonečné vzdálenosti. V případě, že bychom chtěli teoreticky popsat případ, kdy je látce umožněno difundovat pouze v omezené oblasti, musíme postupovat následovně.

Představme si, že válec vody, do které bude roztok difundovat, má konečnou délku l a zaveďme si podmínku, že na vrcholu tohoto válce nedochází k difúznímu toku, tedy

$$\partial C / \partial x = 0, \qquad x = l. \tag{1.23}$$

Tato podmínka je splněna, pokud je křivka koncentrace na hranici prostředí zrcadlena a tato zrcadlená křivka je superponována na křivku originální. V námi uvažovaném systému konečných rozměrů to znamená, že křivka v místě $x = l$ je zrcadlena na křivku v místě $x = 0$ a poté v $x = l$ a tak dále - každá následná reflexe je superponována na originální křivku (1.22). Jelikož je originální řešení součtem dvou chybových funkcí, kompletní výraz pro vývoj koncentrace v prostředí s konečnými rozměry rovno neko-

nečné sérii součtů chybových funkcí, či jejich komplementů, tudíž dostaneme

$$\begin{aligned} C = \frac{1}{2}C_0 \Bigg\{ & \operatorname{erfc}\frac{x-h}{2\sqrt{Dt}} - \operatorname{erfc}\frac{x+h}{2\sqrt{Dt}} + \operatorname{erfc}\frac{2l-h-x}{2\sqrt{Dt}} \\ & - \operatorname{erfc}\frac{2l+h-x}{2\sqrt{Dt}} + \operatorname{erfc}\frac{2l-h+x}{2\sqrt{Dt}} - \operatorname{erfc}\frac{2l+h+x}{2\sqrt{Dt}} \\ & + \operatorname{erfc}\frac{4l-h-x}{2\sqrt{Dt}} - \operatorname{erfc}\frac{4l+h-x}{2\sqrt{Dt}} + \dots \Bigg\} \\ = \frac{1}{2}C_0 & \sum_{n=-\infty}^{\infty} \left\{ \operatorname{erf}\frac{h+2nl-x}{2\sqrt{Dt}} + \operatorname{erf}\frac{h-2nl+x}{2\sqrt{Dt}} \right\}. \end{aligned} \tag{1.24}$$

2. Celulární automaty

Celulární automaty jsou diskrétním systémem, který je složený z jednotlivých buněk, kdy je, při jednotlivých krocích evoluce, na každou z těchto buněk simultánně aplikováno evoluční pravidlo.

2.1 Historie

S myšlenkou sestrojit celulární automaty jako první přišel John von Neumann. Bylo to v době kdy pracoval na výrobě prvního digitálního počítače (40-tá léta 20. století) a jeho koncept celulárních automatů vznikl při jeho snaze najít a zkonstruovat systém, který by byl schopen sám sebe reprodukovat. Jeho vidinou byly počítače, které by byly schopny se samy opravovat. To ho vedlo k vytvoření diskrétního světa složeného z buněk (odtud název celulární automaty), ve kterém aktivity probíhají simultánně (analogie s biologickým systémem).

Jím zkonstruovaný (sebereplikující se) celulární automat představoval dvojrozměrnou čtvercovou mřížku, která obsahovala až několik tisíc buněk a každá buňka mohla nabývat jeden z 29 možných stavů.

Velký zájem o celulární automaty podnítil až v roce 1970 matematik John Conway když zformuloval, nyní již velmi známé, pravidlo pro dvojrozměrné celulární automaty, které nazval *Game of Life*. Navázal tak na von Neumannovy poznatky a pokusil se nalézt jednoduché pravidlo, které by vedlo na komplexní chování [8]. Výsledkem byl systém, který teoreticky modeloval chování buněčného systému.

Stanovil, že buňky mohou mít dva stavy – buňka může být buď živá, nebo mrtvá. Evoluční pravidla, podle kterých se *Game of Life* vyvíjí, zní následovně:

1. každá živá buňka, která má méně než dva živé sousedy umírá,

2. každá živá buňka, která má dva nebo tři živé sousedy zůstává naživu,

3. každá živá buňka, která má více než tři živé sousedy zemře,

4. každá mrtvá buňka, která má přesně tři živé sousedy se stane živou.

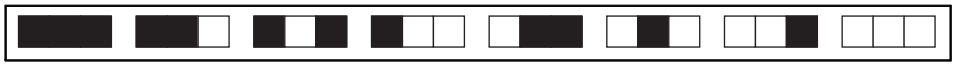
Ačkoliv jsou tato pravidla velmi jednoduchá, dokázala vytvořit velmi rozmanité druhy chování, jejichž studiu se do dnešní doby věnovala velká spousta lidí.

Na začátku 80. let minulého století se Stephen Wolfram věnoval studiu základních pravidel pro základní celulární automaty, které nazval Elementary Cellular Automata (zkráceně ECA) [8, 9]. Wolfram pozoroval, že celulární automaty, ač to jsou diskrétní systémy, vykazují mnohá chování, která jsou známá ze spojitých systémů. Tyto poznatky ho vedly ke zformulování revoluční teorie, kterou popsal v knize *A New Kind of Science* [9].

2.2 Architektura

U celulárních automatů se vždy pracuje s mřížkou buněk, kde každá z těchto buněk může nabývat přesně definovaných hodnot. Nejjednodušší celulární automaty nabývají dvou hodnot, tak jak je tomu kupříkladu v *Game of Life*. Při každém evolučním kroku se všechny buňky v mřížce vyvíjí simultánně podle pravidla, které se mu definuje. V této diplomové práci pracujeme se značením pravidel tak, jak je uvádí ve své knize Wolfram [9]. Ukážeme si to na příkladu elementárních celulárních automatů.

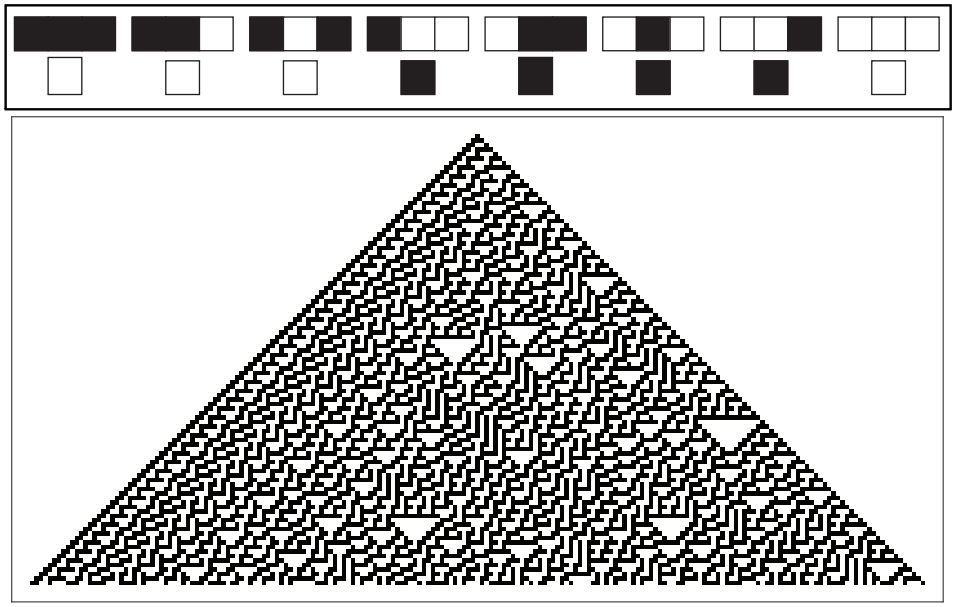
Elementární celulární automaty jsou jednorozměrné, každá buňka může mít pouze dva sousedy a mohou mít pouze dvě hodnoty 0 a 1. Z kombinatoriky víme, že maximální počet stavů, kterých můžou takto definované elementární automaty nabývat, je 8. Každý z těchto osmi stavů mění hodnotu prostřední buňky buď na nulu (bílá) či na



Obrázek 2.1: všechny možné stavy ECA

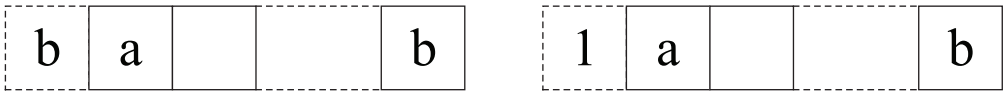
jedničku (černá) v závislosti na tvaru příslušného pravidla. To nás přivádí k otázce, jaký je maximální počet možných pravidel pro elementární celulární automaty. Odpověď na tuto otázku získáme, pokud provedeme variaci s opakováním tak, že chceme získat osm prvků kombinací/variací našich dvou hodnot - bílá a černá. Výsledné číslo je 256. Z toho je vidět, že elementární celulární automaty jsou analogické s osmibitovým systémem.

Wolfram poté zkoumal všechna tato možná pravidla, která poté podrobně popsal ve své knize [9]. Na Obr. 2.2 je znázorněno jedno z Wolframových oblíbených pravidel, rule 30, které se mimochodem používá v softwaru *Mathematica* při generování náhodných čísel.



Obrázek 2.2: rule 30

Další věcí, kterou je potřeba u celulárních automatů zmínit je definování jejich hranic. Každý celulární automat představuje matici buněk a tyto buňky se mění na základě svých sousedů. U Moorova sousedství pro dvourozměrné celulární automaty by měla mít každá buňka 8 sousedů. To platí pro ty co jsou ve středu, ale co ty buňky, které jsou na okrajích této matice buněk? Ty nemůžou mít všech 8 sousedů, ledaže ošetříme hraniční podmínky. Je několik typů hraničních podmínek, které můžeme použít [8]. My jsme se v této diplomové práci zaměřili na 2 typy hraničních podmínek: propojená nebo také periodická plocha a uzavřená nebo také fixovaná plocha (viz Obr.2.3).

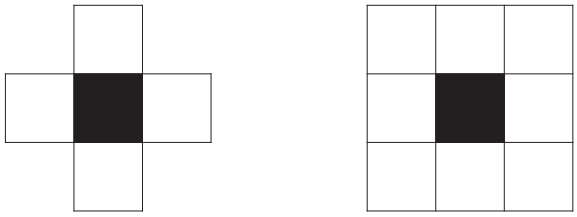


(a) Periodická (b) Fixovaná

Obrázek 2.3: Hraniční podmínky

2.3 Blokové celulární automaty

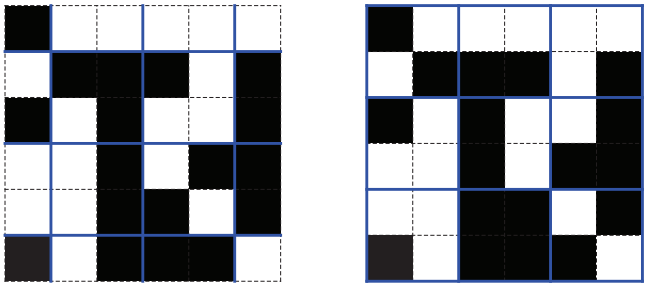
Blokové celulární automaty (dále jen BCA) se svou definicí sousedství diametrálně odlišují od ostatních celulárních automatů využívajících von Neumannovo či Moorovo sousedství (viz Obr. 2.4). U nich se totiž celá mřížka rozdělí na jednotlivé buňky a ke



(a) von Neumann (b) Moore

Obrázek 2.4: Dva typy sousedství pro 2D celulární automaty

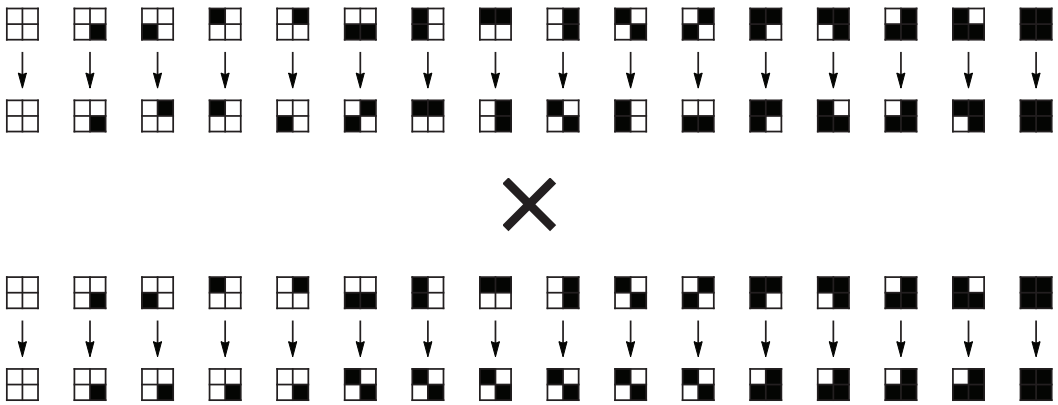
změně těchto buněk dochází na základě jejich sousedů. Naopak u BCA se mřížka buněk rozpadne na malé bloky, které obsahují buď dvě (jednorozměrný případ) či čtyři buňky (dvojrozměrný případ). S myšlenkou blokových celulárních automatů přišel Margolius a proto se také často nazývají Margoliovo sousedství. Ke změně dochází čistě jen uvnitř těchto bloků a tato změna není nijak ovlivněna jejich okolím. Z toho plyne, že by zde nedocházelo k přenosu informace mezi jednotlivými bloky. Jelikož by takto formulované BCA neměly valného užitku, zavádí se posun mřížky, čímž dostaneme liché a sudé kroky evoluce (viz Obr. 2.5). Nyní už je zde přenos informace přítomen.



(a) Lichý krok (b) Sudý krok

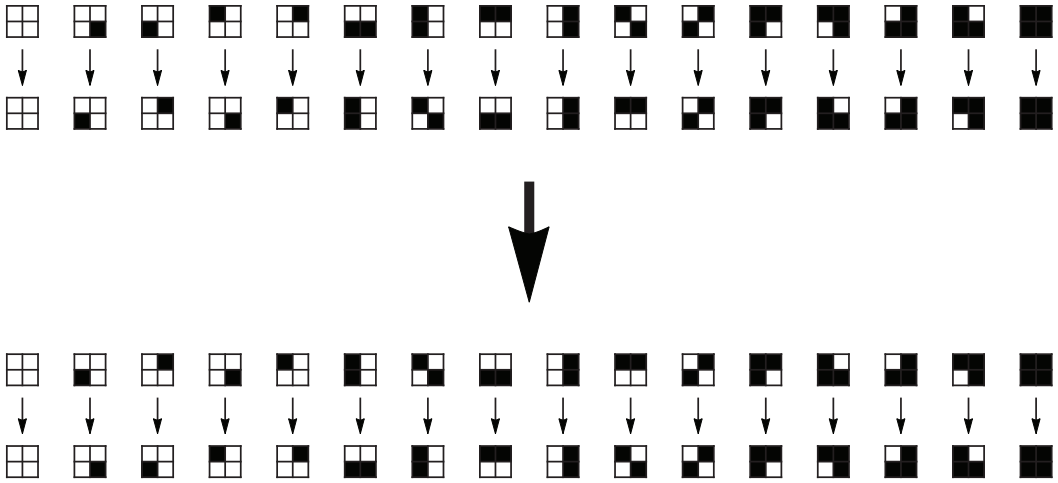
Obrázek 2.5: Ukázka lichého a sudého kroku

Jednou ze zajímavých vlastností BCA je to, že jsou samy o sobě reverzibilní. Samozřejmě musíme dodržet podmínku, že zvolené pravidlo nemůže převést více stavů na jeden stejný (viz Obr. 2.6).



Obrázek 2.6: Ukázka reverzibilního a nereverzibilního pravidla

Jsme schopni vytvořit reverzibilní chování i u klasických celulárních automatů, ale tady je potřeba pro jejich evoluční pravidla zavést, kromě jejich závislosti na okolních buňkách, i závislost na stavech buňky v předešlých evolučních krocích. Naopak u BCA toho dosáhneme pouhým otočením evolučního pravidla (viz Obr.2.7).



Obrázek 2.7: Otočeni pravidla pro vratný děj

3. Různé přístupy k simulování difúze celulárními automaty

Mechanismus celulárních automatů byl objeven již před půl stoletím a tudíž je celkem rozumné předpokládat, že již do tohoto dne vzniklo pár studií, které se nutně musely zabývat myšlenkou studia a simulace difúze podle výše zmíněných celulárních automatů. V této kapitole bychom se chtěli právě těmto studiím blíže věnovat.

V literatuře, která se zabývala touto problematikou [8, 10, 11], jsme našli zmínky o dvou typech celulárních automatů, které se pro simulování difúze dosud používaly. Lišily se typem sousedství, jeden přístup využíval von Neumannova sousedství a druhý využíval sousedství Margoliovo, což je jiný způsob označení blokových celulárních automatů, kterými jsme se zabývali i v naší diplomové práci.

V případě von Neumannova sousedství fungovala evoluční pravidla tak, že pro každou buňku existovala 25% pravděpodobnost, že si tato buňka vymění pozice s jedním se svých sousedů.

Pro případ blokových celulárních automatů používali pouze pravidlo, které říkalo, že pro každý evoluční krok existuje pravděpodobnost p, že dojde k rotaci obsahu každého bloku o $\pi/2$ nebo o $-\pi/2$. Při studiu tohoto pravidla jsme dospěli k názoru, že musí existovat vhodnější pravidla pro simulování difúze pomocí blokových celulárních automatů, než-li pouhé rotace v bloku.

4. Sestavení BCA generátoru

4.1 BCA pomocí NestList

Software *Mathematica* [12, 13] je velmi sofistikované programovací prostředí, které je téměř každým rokem zdokonalováno. Obsahuje dokonce již i předdefinované funkce pro generování celulárních automatů jako je funkce CellularAutomaton. Bohužel, jelikož se blokové celulární automaty se svou definicí od klasických automatů odlišují a také proto, že nejsou tak často používané, v *Mathematice* zatím funkce pro jejich realizaci obsažena není. Tudíž jsme je byli nuceni nadefinovat sami. Výsledný kód vypadá následovně:

```
BCAList[matrix_, i_, j_, k_, steps_] :=
 Module[{n = True},
  NestList[If[n == True,
     n = ! n; Flatten[Partition[#, {2, 2}] /.
       SetRule[{oneCPerm[[i]], twoCPerm[[j]], threeCPerm[[k]]}],
      {{1, 3}, {2, 4}}], n = ! n;
     RotateLeft[
      Flatten[Partition[RotateRight[#, {1, 1}], {2, 2}] /.
        SetRule[{oneCPerm[[i]], twoCPerm[[j]], threeCPerm[[k]]}],
       {{1, 3}, {2, 4}}], {1, 1}]] &, matrix, steps]
]
```

Dříve, než přejdeme k popisu konstrukce funkce pro generování blokových celulárních automatů, měli bychom se okrajově zmínit o způsobu programování v prostředí *Mathematica*.

Mathematica má velmi silnou základnu pro funkční programování. Hranaté závorky ohraničují argumenty funkcí, složenými závorkami se píší vektory a matice, a také představují seznamy prvků a veličin. Jednotlivé argumenty či veličiny se oddělují čárkou, středník se používá pro vložení více prvků do jednoho argumentu.

Nyní můžeme přistoupit k popisu samotné funkce pro blokové celulární automaty.

Jak již bylo v podkapitole 1.3 zmíněno, fungují tak, že na matici buněk, které představují počáteční nadefinování, aplikujeme mřížku, která je rozdělí na stejně velké bloky o čtyřech buňkách. Při evoluci těchto automatů rozlišujeme mezi lichými a sudými kroky, kdy se rozdělovací mřížka posune vždy o jednu buňku, jak je znázorněno na Obr. 2.5. Z těchto poznatků vyplývá, že potřebujeme vytvořit funkci, která si vezme námi nadefinovanou matici a při každém evolučním kroku, u kterých bude rozlišovat zda-li je lichý či sudý, rozdělí matici buněk na řadu bloků o rozměrech 2×2 buňky a každý z těchto bloků se musí změnit podle přesně definovaného evolučního pravidla, které se během evoluce nemění.

Naši funkci jsme nazvali BCAList a jejími argumenty jsou: počáteční matice, kódované značení pravidel a nakonec počet evolučních kroků, po které má evoluce probíhat. Jádrem je funkce Partition, která umí rozdělovat skupiny prvků na námi zvolené menší skupiny - pomocí ní rozdělíme matici buněk na jednotlivé bloky.

Nyní můžeme aplikovat pravidlo evoluce. Zde přichází na řadu další funkce, ReplaceAll, která má symbolický tvar "/.". Do prvního argumentu této funkce se vkládá uspořádaná n-tice prvků, se kterým chceme pracovat a druhým argumentem je skupina pravidel, podle kterých chceme, aby se námi vybrané prvky seznamu změnily - definují se následovně: *původní prvek→nový prvek*. V našem případě je prvním argumentem řada bloků a ve druhém argumentu jsou transformační pravidla, která představují pravidla pro evoluci automatů.

Funkce SetRule byla další funkce, kterou jsme si sami nadefinovali. Obsahuje jeden argument, kterým je List, který se také značí složenými závorkami, o třech neznámých: oneCPerm, twoCPerm, threeCPerm. O těchto proměnných a o samotné funkci SetRule se více zmíníme v následující podkapitole. Nyní jen zmíníme, že představují definici pro evoluční pravidla.

Po aplikaci evolučních pravidel dostáváme řadu bloků, jejichž hodnoty se změnily v závislosti na příslušných pravidel. Jelikož ve výsledku chceme dostat matici buněk, a ne jen řadu bloků, musíme je dát opět dohromady a k tomu použijeme další z mnoha funkcí programu *Mathematica* pro práci s řadami či seznamy proměnných a tou je Flatten. Jejím prvním argumentem je naše řada bloků a druhým argumentem, který bývá volitelný, je kódovaná zpráva, která Flatten sdělí, jakým způsobem je složit dohromady. V našem případě spojí všechny bloky zpátky na matici buněk. Takto jsme vykonali jeden krok evoluce. V tomto případě se jednalo o lichý krok.

Rozdělování matice na jednotlivé bloky je o poznání jednodušší při lichých krocích, nežli u sudých kroků. U těch totiž musíme ještě počítat s tím, že je třeba posunout mřížku, která rozděluje matici buněk na bloky, jak je znázorněno na Obr. 2.5 v podkapitole 1.2.3.

Kód pro evoluci sudého kroku je takřka totožný s kódem pro krok lichý, opět používáme Partition a ReplaceAll pouze s jediným rozdílem, kterým je přidání funkce RotateRight, jež aplikujeme na matici buněk dříve, nežli je rozdělíme na jednotlivé bloky. Tato funkce je stěžejní pro evoluci blokových celulárních automatů a umožňuje nám chytrým a nenáročným způsobem provést pomyslný posun rozdělovací mřížky. Její využití si ukážeme na následujícím příkladě.

Mějme matici buněk o rozměrech 6×6 a každou buňku si pro přehlednost označme číslem od 1 do 36 (Obr. 4.1).

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

Obrázek 4.1: Počáteční matice buněk

Při sudém kroku evoluce bude rozdělovací mřížka umístěna tak, jak je ilustrováno na Obr. 4.2. Hned na první pohled je vidět, že na okrajích máme neúplné bloky, ale to napravíme, pokud aplikujeme hraniční podmínku. V našem případě jsme všechny strany vzájemně propojili a získali jsme něco na způsob toroidu (Obr. 4.3).

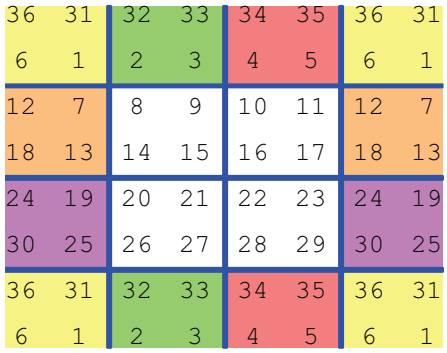
1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

Obrázek 4.2: Rozdělovací mřížka

Je však zbytečné přidávat na všechny strany dodatečné buňky, jejichž hodnoty budou stejné (Obr. 4.4). Nám stačí jen přesunout pravou hranu doleva a spodní hranu nahoru. To jsme uskutečnili pomocí již zmiňované funkce RotateRight. Jejím prvním

36	31	32	33	34	35	36	31
6	1	2	3	4	5	6	1
12	7	8	9	10	11	12	7
18	13	14	15	16	17	18	13
24	19	20	21	22	23	24	19
30	25	26	27	28	29	30	25
36	31	32	33	34	35	36	31
6	1	2	3	4	5	6	1

Obrázek 4.3: Rozšíření matice



Obrázek 4.4: Rozšíření matice 2

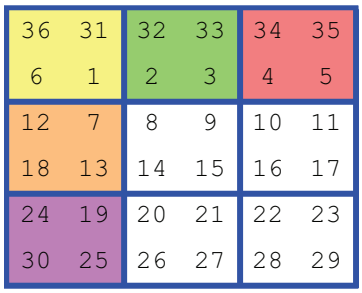
argumentem je požadovaná matice a ve druhém argumentu udáváme způsob rotace (Obr. 4.5).

36	31	32	33	34	35
6	1	2	3	4	5
12	7	8	9	10	11
18	13	14	15	16	17
24	19	20	21	22	23
30	25	26	27	28	29

Obrázek 4.5: Použití RotateRight

Dostali jsme tak matici, která obsahuje všechny potřebné bloky aniž by se nám některý vyskytoval vícekrát (Obr. 4.6). Po aplikování evolučního pravidla na takto vzniknuvší matici není pro nás nic snazšího nežli vrátit buňky do původního uspořádání funkcí RotateLeft, u které budeme mít stejný argument pro rotaci jako u RotateRight.

Jakmile jsme měli hotový mechanismus pro evoluci lichých a sudých kroků, vložili jsme jej do funkce NestList a opatřili ho spínačem v podobě proměnné n, kterou jsme pomocí Module definovali jako lokální proměnnou. Funkce Nest obaluje svůj druhý



Obrázek 4.6: Použití RotateRight 2

argument prvním argumentem s tím, že ji ve třetím argumentu zadáme, kolikrát tak má učinit. V případě, že první argument je f a druhý argument je x, tak pokud zadáme, ať se provede kupříkladu třikrát, dostaneme $f[f[f[x]]]$. My jsme použili funkci NestList, která nám vytvoří uspořádanou n-tici všech vnoření, tedy, všech kroků evoluce.

4.2 Pravidla pro evoluci BCA

S hotovým mechanismem pro generování průběhu BCA, jsme se mohli pustit do hledání pravidla, které by nejlépe splňovalo teoretický model difúze. Nejdříve jsme ale museli stanovit, která pravidla budeme zkoumat.

Z kapitoly 2.3 víme, že u dvourozměrných BCA může každý blok nabývat jednu ze 16 možných hodnot. Každé evoluční pravidlo vždy obsahuje přesně definovaný způsob evoluce pro každý z možných stavů celulárního automatu. Pro dvojrozměrné BCA to dává celkem 16^{16} možných pravidel. Což představuje až příliš velký počet možností. Naštěstí jsme byli schopni toto číslo zmenšit díky předpokladu, že by měla být v celém systému zachován počet částic a také by v tomto systému nemělo docházet ke ztrátě informace, což znamená, že by se žádné dva rozdílné bloky neměly transformovat na blok se stejným uspořádáním. To nám umožnilo zmenšit astronomické číslo 16^{16} na mnohem snesitelnější velikost, a to na 414 720 pravidel. Rozdělili jsme tedy 16 stavů BCA do pěti skupin podle počtu buněk a z nich jsme poté skládali pravidla. Krajní stavy, kdy blok je buď prázdný nebo plný, mají jen jednu možnost změny, a to na sebe sama. Pokud má blok jen jednu, nebo tři buňky, může tato buňka, či buňky, může zaujímat čtyři možné stavy. Celkový počet možných evolucí, které vyhovují našim podmínkám, dostaneme jako 4!, což dává 24 možných evolucí pro každý z těchto dvou případů. Poslední možnost je ta, že se v bloku budou vyskytovat přesně dvě buňky. V tomto

případě můžou v bloku zaujímat přesně šest stavů, což dává celkem 720 (6!) možných pravidel evoluce. Zkombinováním všech pravidel pro jednu, dvě a tři buňky jsme dostali jsme získali již zmíněné číslo 414 720.

K rozlišení jednotlivých pravidel, která splňují výše uvedené podmínky, jsme si zavedli vlastní způsob značení, který sestává ze tří cifer, které oddělujeme pomlčkou. Každá z těchto cifer představuje pravidlo pro určitý počet buněk a my toto značení čteme následovně: *pravidlo pro 1 buňku–pravidlo pro 2 buňky–pravidlo pro 3 buňky.* Vzorové pravidlo by tedy vypadalo takto: 5-250-6. To nás přivádí k funkci SetRule, kterou jsme vytvořili pro generování podmínek evoluce pro jednotlivá pravidla.

```
SetRule[lis_] :=
 MapThread[Rule,
  {Join[{{{0, 0}, {0, 0}}}, oneC, twoC, threeC, {{{1, 1}, {1, 1}}}],
   Join[{{{0, 0}, {0, 0}}}, Flatten[lis, 1], {{{1, 1}, {1, 1}}}]}]
```

Nejdříve popíšeme proměnné oneCPerm, twoCPerm a threeCPerm. Jsou to permutace všech možných stavů, které můžou jedna, dvě nebo tři buňky v jednom bloku zaujímat. Označili jsme je proměnnými oneC, twoC a threeC. Jednotlivé proměnné oneCPerm, two-

```
oneC = {{{0, 0}, {0, 1}}, {{0, 0}, {1, 0}}, {{1, 0}, {0, 0}},
   {{0, 1}, {0, 0}}};
twoC = {{{0, 0}, {1, 1}}, {{1, 0}, {1, 0}}, {{1, 1}, {0, 0}},
   {{0, 1}, {0, 1}}, {{1, 0}, {0, 1}}, {{0, 1}, {1, 0}}};
threeC = {{{1, 1}, {1, 0}}, {{1, 1}, {0, 1}}, {{0, 1}, {1, 1}},
   {{1, 0}, {1, 1}}};
oneCPerm = Permutations[oneC];
twoCPerm = Permutations[twoC];
threeCPerm = Permutations[threeC];
```

CPerm a threeCPerm tvoří tedy uspořádané n-tice jednotlivých možností a my je jsme schopni jednotlivě volat pomocí funkce Part, jejíž první argument představuje daný seznam prvků, a druhý argument udává pořadí či umístění prvku, který chceme získat. My hlavně používáme jejího symbolického vyjádření, které je mnohem jednodušší na zápis. Funguje tak, že za námi zkoumaný seznam prvků vložíme dvě skupiny hranatých závorek, které budou obsahovat pořadí požadovaného prvku. Pokud bychom kupříkladu chtěli použít pro jednu buňku pravidlo 6, výsledný zápis by vypadal takto: oneCPerm[[6]].

Funkce SetRule pomocí MapThread vytvoří příkazy pro změnu bloků tak, že propojí první seznam, který zahrnuje všech šestnáct možných stavů, s druhým seznamem,

který zase představuje stavy, na které se mají při evoluci změnit.

4.3 Využití cyklu Do

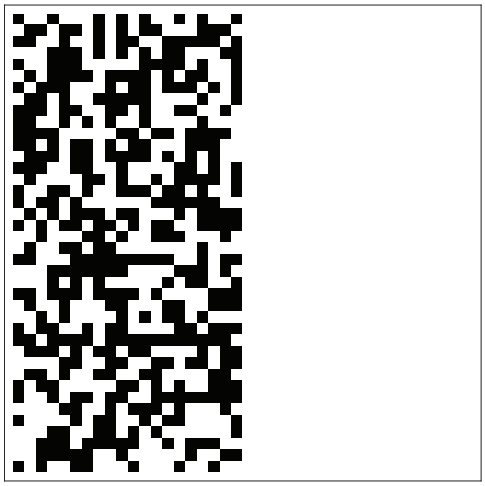
Když jsme prováděli evoluci na matici buněk o velkých rozměrech, 500 a výše, po určitém množství kroků/iterací jsme narazili na limity, které funkce NestList má. Každý krok evoluce totiž se nabaluje na ty předešlé, a jelikož z nich tvoří řadu, narážíme zde na limity operační paměti, která po jednom až dvou tisících krocích vypovídá služby a celý výpočet se stornuje a my ztrácíme potřebná data. Tudíž jsme přišli s dalším způsobem generování BCA, a to s využitím cyklu Do. Využíváme ho v případech, kdy nepotřebujeme zaznamenat každý krok, ale stačí nám provádět analýzu evolučního průběhu kupříkladu po určitých úsecích. Díky tomu nezatěžujeme paměť počítače, jelikož při každé operaci pracuje pouze s jednou konkrétní maticí.

```
n = True;
Do[
 If[n == True,
  n = ! n;
  mat = Flatten[Partition[mat, {2, 2}] /.
      SetRule[{oneCPerm[[6]], twoCPerm[[666]],
        threeCPerm[[6]]}], {{1, 3}, {2, 4}}],
  n = ! n;
  mat = RotateLeft[
    Flatten[Partition[RotateRight[mat, {1, 1}],
      {2, 2}] /.
     SetRule[{oneCPerm[[6]], twoCPerm[[666]],
       threeCPerm[[6]]}], {{1, 3}, {2, 4}}], {1, 1}]],
 {X}]
```

5. Prvotní roztřídění

Jakmile jsme měli definována pravidla, která budeme zkoumat a měli jsme sestrojený mechanismus pro jejich evoluční průběh, mohli jsme přejít k samotnému cíli této diplomové práce, a tím je najít pravidlo pro BCA, které by nejvíce odpovídalo difúznímu průběhu.

Postupovali jsme tak, že jsme nejdříve chtěli od oddělit pravidla, jejichž průběh z nich dělal možné kandidáty pro simulování difúze, od pravidel, jejichž průběh se k simulování difúzního chování použít nedal. Týkalo se to těch pravidel, jejichž evoluce představovala příliš rychlé fluktuace částic, a nebo naopak se částice takřka nehýbaly. Toto třídění jsme realizovali tak, že jsme si vytvořili počáteční matici, kterou jsme vertikálně rozdělili na dvě rovnoměrné půlky, kde levou půlku jsme zaplnili náhodnou distribucí částic a pravou jsme nechali prázdnou. Měřili jsme počet částic v obou těchto



Obrázek 5.1: Počáteční definice matice

částech v průběhu evoluce.

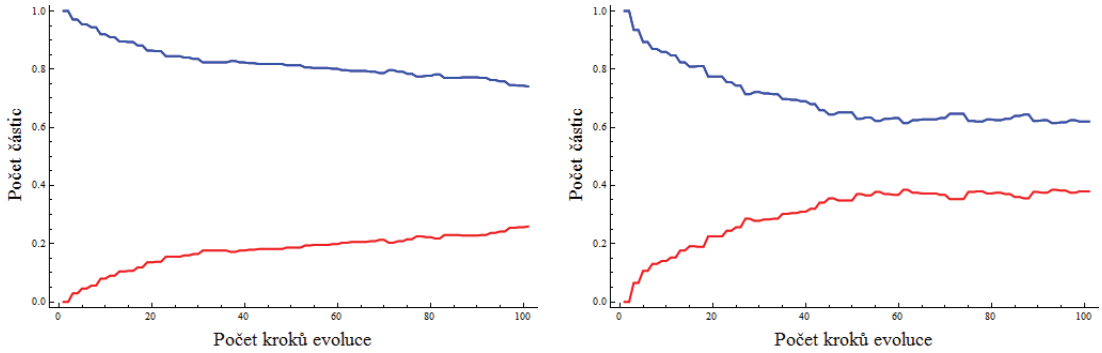
Vycházeli jsme z předpokladu, že při difúzním chování budou mít částice tendenci difundovat z míst s vysokým gradientem koncentrace (levá strana) do míst, kde je tento gradient nižší, tudíž se budou pohybovat směrem do pravé, prázdné, strany až se

hodnota koncentrace v celém prostoru vyrovná na stejnou hodnotu. Když tedy budeme měřit počet částic na levé a pravé straně a vyneseme je do grafu, kde na ose x bude doba evoluce v evolučních krocích, dostaneme dvě křivky, jedna začne na maximální hodnotě, druhá na nulové, které by se měly vzájemně rovnoměrně přibližovat ke střední hodnotě a dále již neklesat, či nerůst.

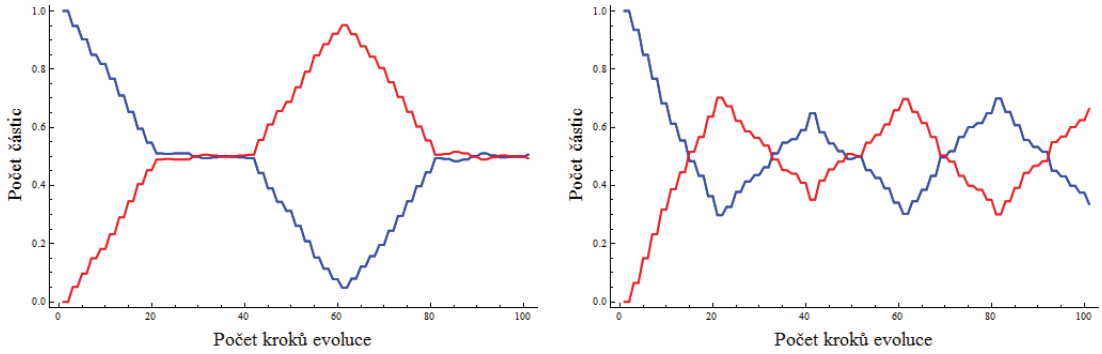
Podmínky pro toto třídění jsme se snažili definovat tak, abychom získali co možná nejsmysluplnější informaci o průběhu konkrétního pravidla, za co možná nejkratší dobu průběhu. Vedla nás k tomu skutečnost, že když by výpočet průběhu každého pravidla zabral byť jen sekundu reálného času, tak při celkovém počtu 414 720 pravidel bychom museli nechat výpočty probíhat po několik dní v kuse. Proto jsme pracovali s plochou o rozměrech 40×40 buněk, jelikož s rostoucí plochou matice roste i doba výpočtu, a evoluce vždy trvala jen 100 kroků. Tato doba evoluce se může jevit příliš krátká, ale ve skutečnosti nám tento poměr plochy matice ku době průběhu stačil pro odseparování pravidel, jejichž evoluční průběh pro nás neměl valného užitku. Jak jsme již dříve zmínili, byli jsme schopni identifikovat tři typy průběhů jednotlivých pravidel. Prvním byl průběh, který jsme hledali – dvě křivky konvergující ke střední hodnotě populace (Obr. 5.2). Dalším byl průběh, kdy obě křivky fluktuovaly okolo střední hodnoty s příliš vysokou frekvencí na to, aby dotyčná pravidla vykonávala difúzní proces (Obr. 5.3) a na konec to byl typ průběhu, jehož částice se takřka nehýbaly a obě křivky zůstávaly (relativně) konstantní (Obr. 5.4).

Jelikož jsme věděli, jaké typy průběhů můžeme očekávat, kód, který se staral o třídění jednotlivých pravidel, byl definován tak, že roztřiďoval podle průběhu spodní křivky, jejíž gradient koncentrace je vždy na počátku nulový. Použili jsme k tomu funkci Which, která vezme zkoumaný prvek a postupně ho porovná se všemi zadanými možnostmi a první, která vrátí hodnotu True se provede. První podmínkou, kterou náš kód zkoumal, byl případ, kdy dané pravidlo způsobovalo kmitání obou křivek kolem střední hodnoty (Obr. 5.3). Tento případ jsme si označili pracovním názvem „vysoké fluktuace“.

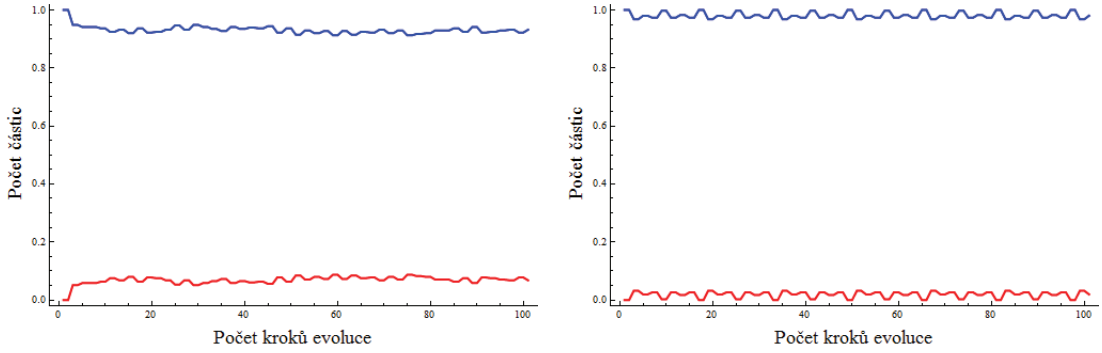
Pomocí funkce Max jsme našli maximální hodnotu spodní křivky a v případě, že tato hodnota byla větší, než 70% z počtu všech buněk v systému, tak potom toto pravidlo vykazovalo průběh s příliš vysokými fluktuacemi a bylo následovně umístěno do patřičné složky. V případě, že dané pravidlo tuto podmínku nesplňovalo, funkce Which se přesunula na další podmínku v pořadí a tato konkrétní podmínka zjišťovala, zda-li



Obrázek 5.2: Průběh s možným difúzním charakterem



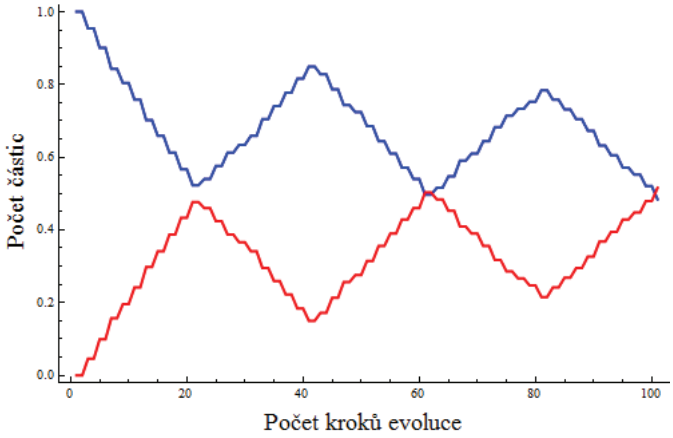
Obrázek 5.3: Průběh s vysokou fluktuací



Obrázek 5.4: Nerostoucí průběh

se jedná o pravidlo s difúzním průběhem. Pro tento případ jsme si definovali proměnnou, kterou jsme nazvali secna. Její hodnotou byl úhel ve stupních, který s x-ovou osou svírala přímka protínající body představující počet buněk na pravé straně matice v desátém a devadesátém kroku evoluce. Pravidlo jsme nechali zařadit do složky obsahující difúzní průběh v případě, že úhel byl větší než 2°. Pokud pravidlo nesplňovalo ani tuto podmínku, nepotřebovali jsme provádět další testy, jelikož už mohlo jít pouze o případ, kdy průběh pravidla zůstává konstantní.

Tato metoda třídění však nebyla stoprocentní, chybu zde totiž zanášela sama hraniční podmínka. Vyskytly se tudíž i průběhy, které, ačkoliv nepředstavovaly námi požadovaný průběh, vlezly se pod požadovanou hranici 70%. Jelikož jsme ale pořizovali vizu-



Obrázek 5.5: Nevyhovující průběh, který přesto prošel selekcí

ální záznam, byli jsme schopni tuto chybu napravit. V Tab.5.1 na následující straně jsou shrnuty výsledky tohoto třídění. Pro větší přehlednost jsme jednotlivé typy průběhů rozdělili do 24 skupin podle pravidla pro jednu buňku, tedy podle pravidel první složky.

Pravidla 1. složky	Nerostoucí	Vysoké fluktuace	Vyhovující průběh
1	963	0	16317
2	70	0	17210
3	623	0	16657
4	0	486	16794
5	0	486	16794
6	17	257	17006
7	62	0	17218
8	0	5185	12095
9	0	480	16800
10	11	0	17269
11	0	5038	12242
12	0	489	16791
13	0	489	16791
14	0	4979	12301
15	18	301	16961
16	0	478	16802
17	0	4748	12532
18	0	282	16998
19	9	0	17271
20	0	477	16803
21	0	479	16801
22	620	0	16660
23	0	273	17007
24	705	0	16575

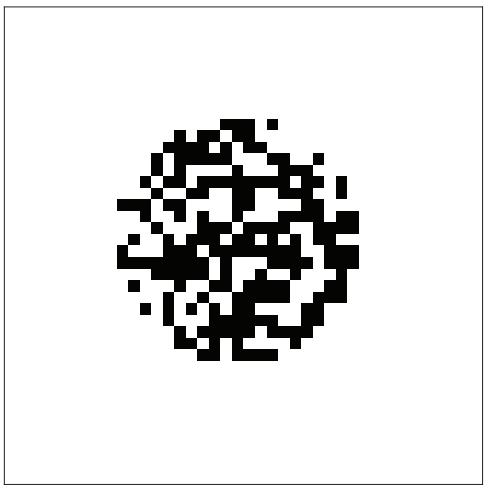
Tabulka 5.1: Výsledné roztřídění pravidel

6. Použití ContourPlot při zkoumání průběhů

Podařilo se nám oddělit pravidla s nepodstatným průběhem od pravidel, která by mohla být kandidáty pro simulování difúze. V tomto okamžiku jsme však nutně museli přijít se způsobem jak podrobněji zkoumat průběh jednotlivých pravidel, která jsme z předešlého třídění získali. Chtěli jsme najít metodu, která by nám dávala informaci o pohybu buněk v prostoru a v čase, pracujeme totiž s dvojrozměrnými BCA. Předešlá metoda pouze sečetla buňky ve velké oblasti, aniž by nějak rozlišovala jejich konkrétní pozice.

Velký problém pro nás představovala sama diskrétní charakteristika BCA a skutečnost, že jsme pracovali s pravidly, kdy buňky mohly nabývat pouze dvou hodnot. Po několika neúspěšných pokusech jsme se nakonec rozhodli rozdělit průběh na krátké úseky, jejichž plochy, což byly matice obsahující jedničky a nuly, jsme sečetli a poté je vynesli pomocí Contour Plot jako vrstevnicový graf. Byli jsme tak schopni vidět místa, kde se buňky v té době nejčastěji vyskytovaly a také jsme byli schopni určit směr jejich šíření. Šlo nám hlavně o to identifikovat směr šíření buněk pro jednotlivá pravidla. Jako počáteční distribuci jsme zvolili náhodně rozmístěné buňky v kruhové ploše (Obr. 6.1). Volili jsme tak z toho důvodu, že jsme chtěli najít pravidla, podle kterých by se buňky rozšiřovali do volného prostoru co nejvíce rovnoměrně a ve všech směrech.

Rozhodli jsme se zjistit, jak moc jednotlivé složky námi definovaných pravidel ovlivňují způsob, jakým se buňky pohybují po ploše. Nejdříve jsme začali se zkoumáním první složky pravidla, což představovalo evoluci bloku s jednou buňkou, a ukázalo se, že jsme zvolili dobře, jelikož jsme zjistili, že největší vliv na průběh daného pravidla má právě tato složka, jak záhy ukážeme. Postupovali jsme tak, že jsme zvolili několik náhodných hodnot pro zbylé dvě složky a na každé z jejich kombinací jsme vyzkoušeli všech 24 možných evolucí první složky. Naše matice měla rozměry 60×60 a počáteční



Obrázek 6.1: Kruhová počáteční distribuce

uspořádání bylo pro všechna použitá pravidla totožné. Částice byly na počátku koncentrovány v kruhu, který měl poloměr 15 buněk a střed měl ve středu matice. Evoluce probíhala po 200 kroků, které jsme rozdělili na menší úseky o délce 10-ti kroků a ty jsme vynesli pomocí ContourPlot.

Když jsme prozkoumali výsledné průběhy, všimli jsme si, že tvar průběhů pravidel, která měla totožnou první složku, Jakmile jsme takto prozkoumali všech 24 možných pravidel pro složku s jednou buňkou, rozdělili jsme je do sedmi skupin v závislosti na směru a způsobu jak se buňky rozmísťovali do prostoru.

První skupina zahrnuje pouze pravidla, jejichž první složka má číslo 1 a jejich průběh vypadá tak, že se buňky takřka nerozmisťují do prostoru, dochází maximálně k přeuspořádání tvaru počáteční distribuce. Takovýto průběh je zobrazen v Obr. 6.2, kde tento průběh odpovídá pravidlu 1-107-17. Je zde zřetelně vidět, že na počátku máme částice nashromážděné v kruhové distribuci a s rostoucím počtem evolučních kroků dochází pouze k přeuspořádání tvaru této kruhové distribuce. Nepozorujeme žádné částice, které by se od této „masy“ oddělovaly.

Nejpočetnější skupinou byla pravidla, u kterých docházelo k pohybu částic pouze ve směru jedné z diagonál. Do této skupiny spadala pravidla, jejichž první složka byla 4, 5, 6, 9, 12, 13, 15, 16, 20 a 21. U složek 4, 5, 6, 12 a 20 se buňky pohybovali po diagonále mezi prvním a třetím kvadrantem a složky 9, 13, 15, 16 a 21 vykonávaly pohyb po diagonále mezi druhým a čtvrtým kvadrantem. Na Obr. 6.3 je vidět průběh pravidla 5-441-17. Je vidět, že již v intervalu 11 až 20 kroků evoluce dochází k rapidnímu posunu částic ve směru diagonály.

Další skupina obsahuje pravidla jejichž první složky mají hodnoty 2, 3, 7 a nebo 22. U těchto pravidel se buňky formují do trojúhelníku, jehož vrchol směřuje pro každou složku do jiného směru: pro složku 2 směřuje vrchol nahoru, pro 3 doleva, pro 7 dolů a pro 22 doprava. Na Obr. 6.4 je ukázka pravidla 7-290-13. Vidíme, že s rostoucím počtem evolučních kroků dochází ve vrchní polovině matice k pohybu částic směrem od středu, ale naopak ve spodní části matice dochází ke shromažďování částic. Již od intervalu 41 až 50 kroků je vidět, že částice se formují do tvaru hrotu trojúhelníku. Z posledních pěti intervalů je znatelné, že jakmile se částice zformují do tvaru hrotu trojúhelníku, tento tvar si již nadále zachovávají a nedochází k jejich rozptylu.

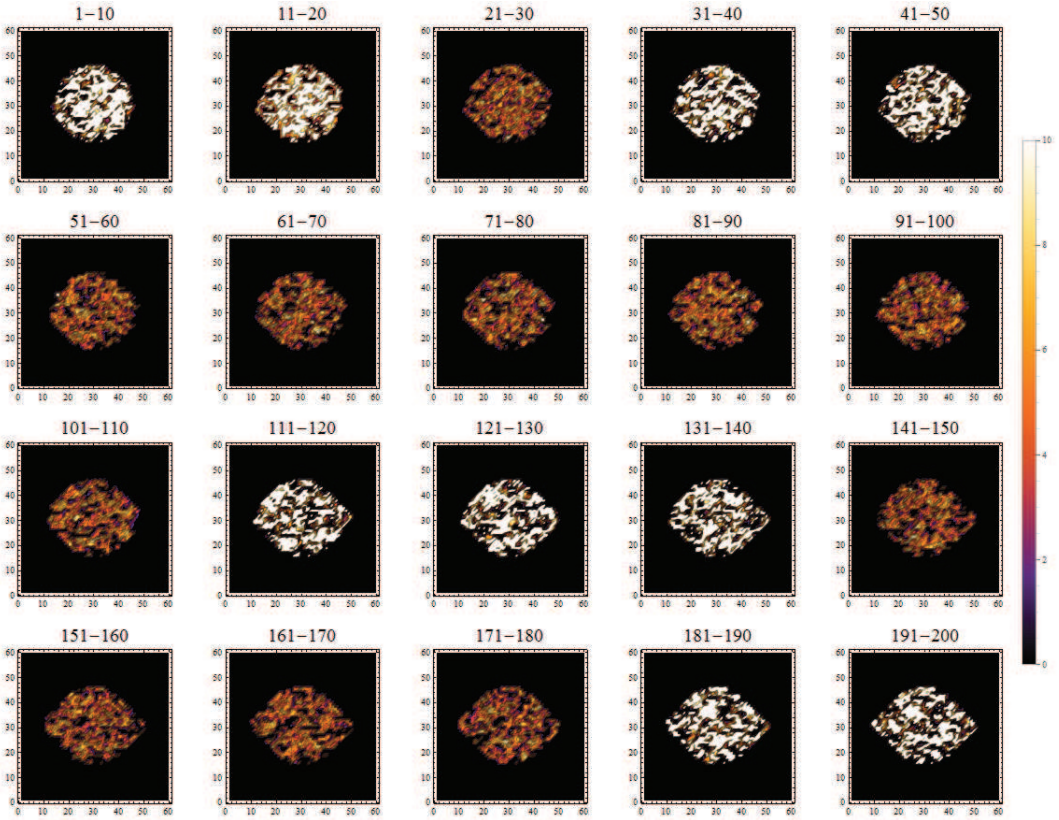
Čtvrtou skupinou byla pravidla se složkami 11, 14, 18 a 23. Pohyb buněk u těchto pravidel vypadal tak, že se z počáteční kruhové distribuce zformovaly tři proudy, které se vydaly každý jiným směrem. Na Obr. 6.5 je vidět průběh pravidla 14-490-18. Ačkoliv je zde, hlavně v pozdějším vývoji, viditelný pohyb částic ve všech směrech, stále se nejedná o pravidla, ze kterých by se mohlo vycházet při simulaci difúzního průběhu.

Pátá skupina je svým průběhem velmi blízká předešlé skupině. Obsahuje jen jednu složku a to číslo 17, a od předešlé skupiny se liší pouze tím, že zde vznikají čtyři proudy místo tří. Průběh jednoho z těchto pravidel, konkrétněji pravidla 17-370-11, je vyobrazen na Obr. 6.6. Na začátku vývoje je viditelné, že se z počáteční, kruhové distribuce částic, začínají částice oddělovat na čtyřech konkrétních místech, která představují průniky diagonál s kruhovou distribucí.

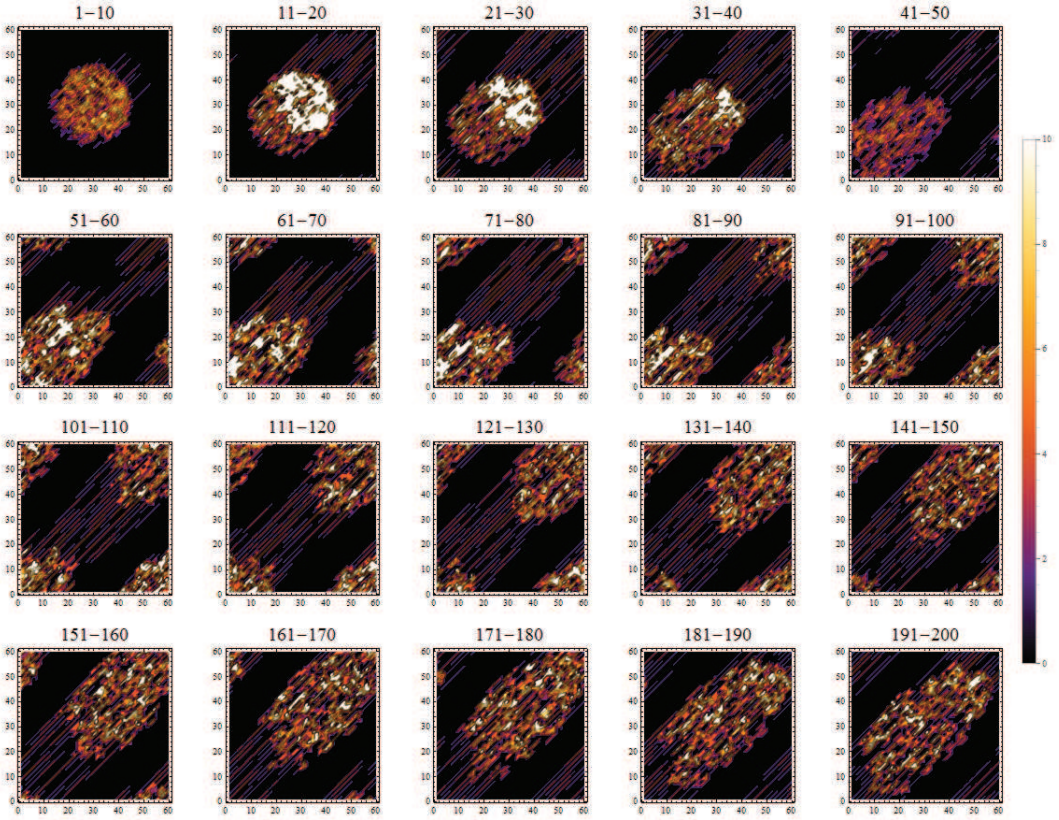
Předposlední skupinou jsou pravidla se složkami 8 a 24. Tato skupina má zajímavý průběh v tom, že se na počátku vývoje začnou částice oddělovat a pohybovat pouze v horizontálním směru (složka s číslem 8) nebo ve vertikálním (složka s číslem 24). Průběh pravidla 8-33-8 je zobrazen na obrázku Obr. 6.7 a je zde zřetelně vidět, že již na začátku evoluce dochází k velmi rychlému pohybu částic směrem od počáteční distribuce, a to pouze v horizontálním směru.

Poslední skupinou pravidel, která nám zbyla, jsou ta, jejichž první složka je buď 10 nebo 19. A právě tato pravidla se ukázala jako jediná, která umožňují částicím naší pomyslné látky, aby se rozptylovaly do prostoru co možná nejrovnoměrněji. Na Obr. 6.8 a Obr. 6.9 je ukázán průběh pravidel 19-4-21 a 10-50-24.

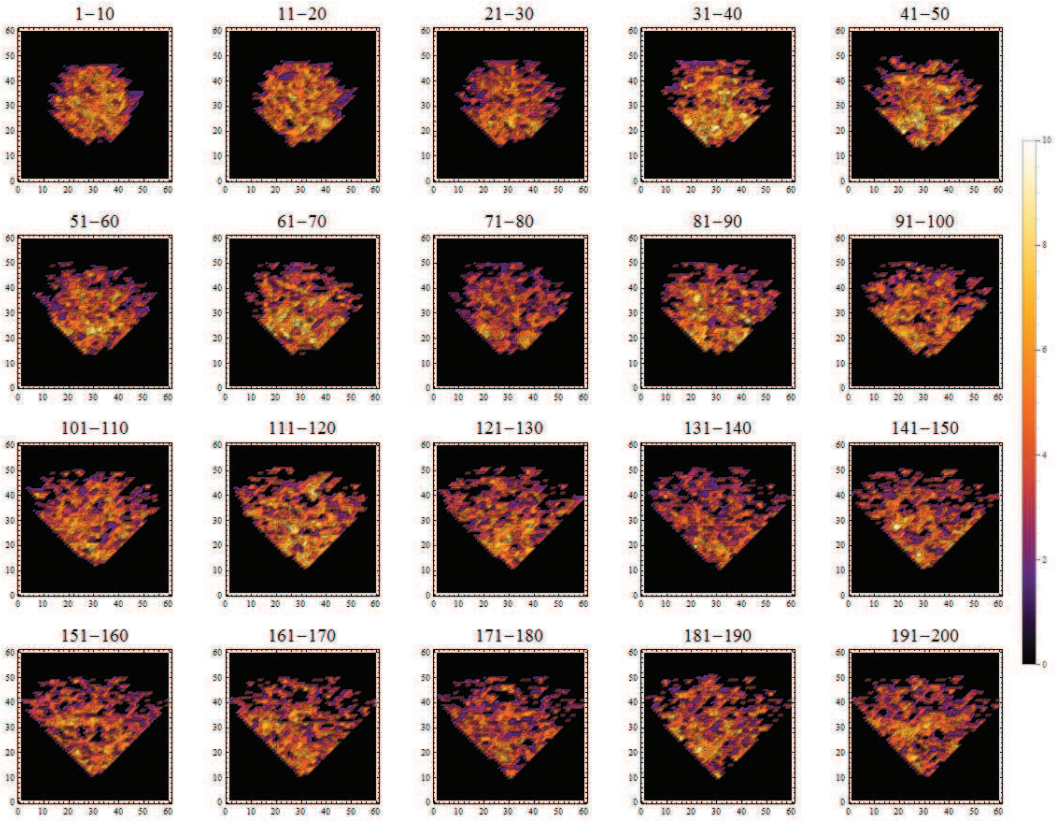
Výsledkem tohoto třídění bylo, že ze 24 možných pravidel pro pohyb/vývoj jedné buňky v bloku, pouze dvě splňují naše požadavky. Když se blíže podíváme na to, jakým způsobem se buňka v bloku pohybuje, pokud je použita jedna z těchto dvou možných



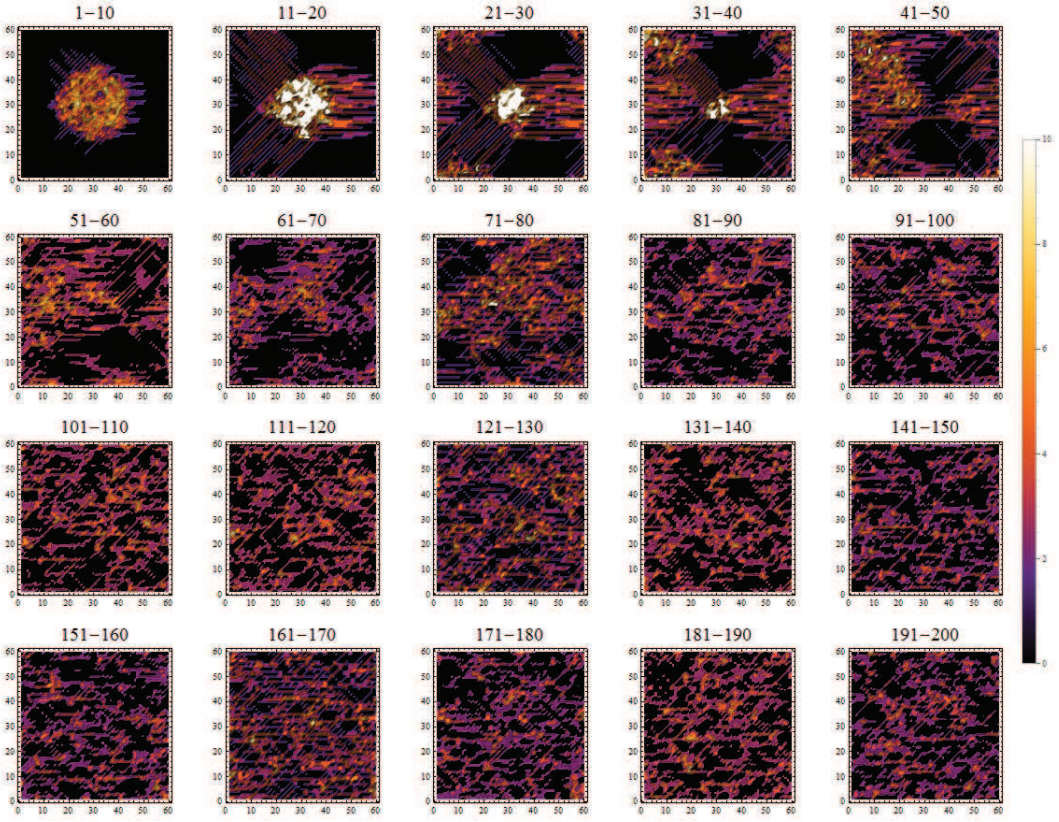
Obrázek 6.2: Ukázka průběhu pravidla z 1.skupiny



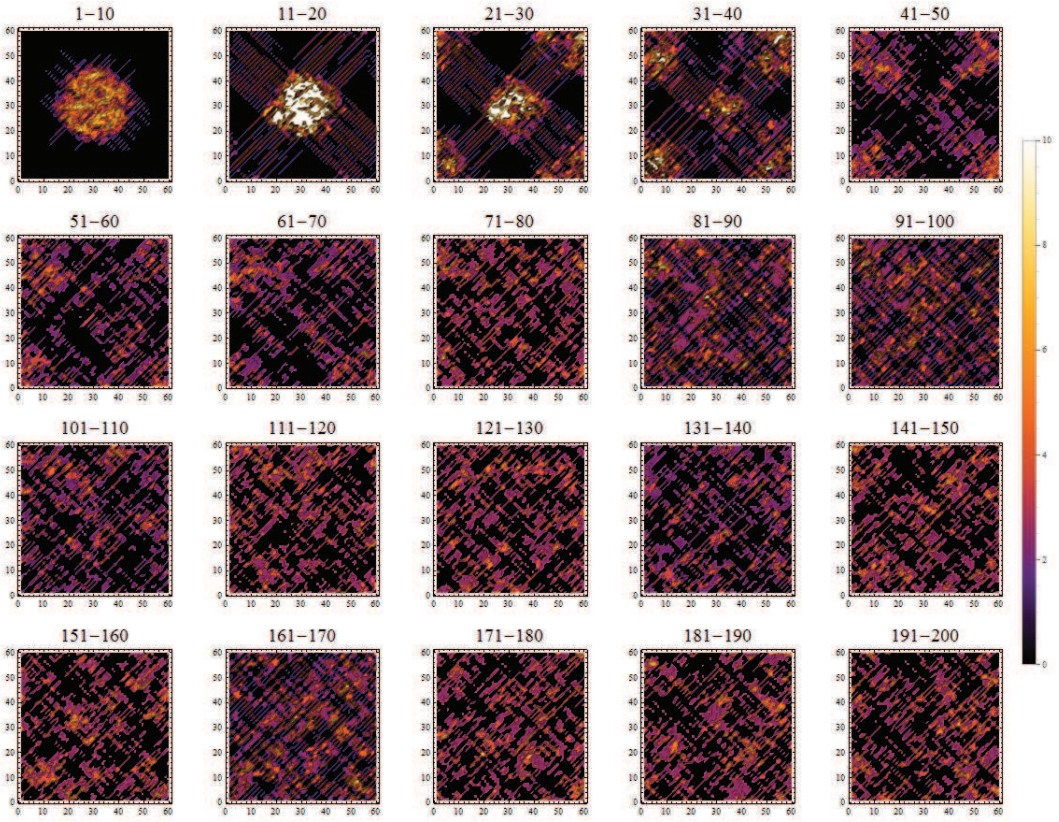
Obrázek 6.3: Ukázka průběhu pravidla z 2.skupiny



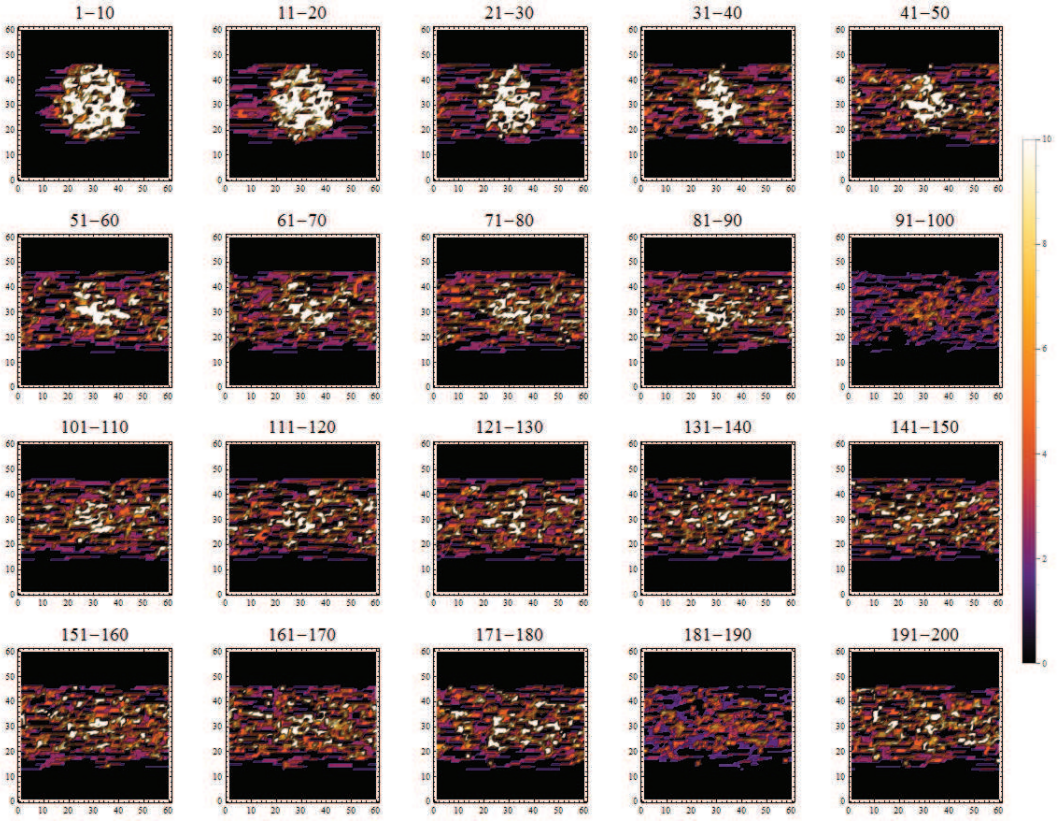
Obrázek 6.4: Ukázka průběhu pravidla ze 3.skupiny



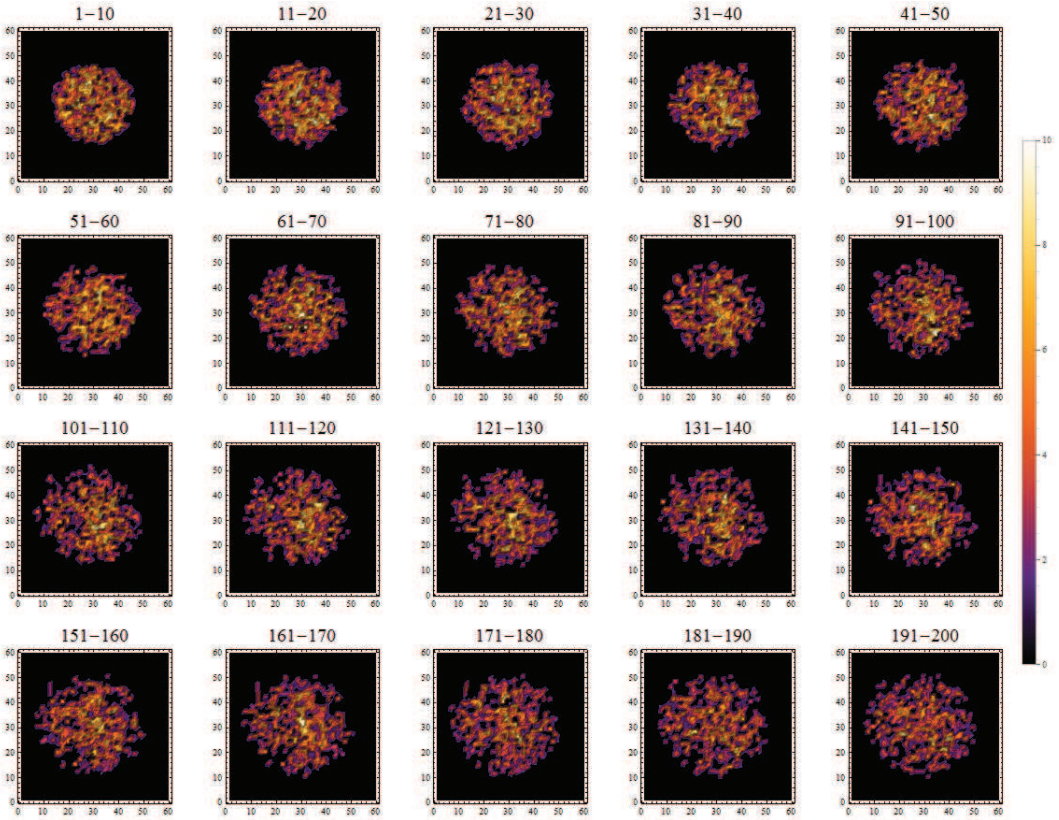
Obrázek 6.5: Ukázka průběhu pravidla ze 4.skupiny



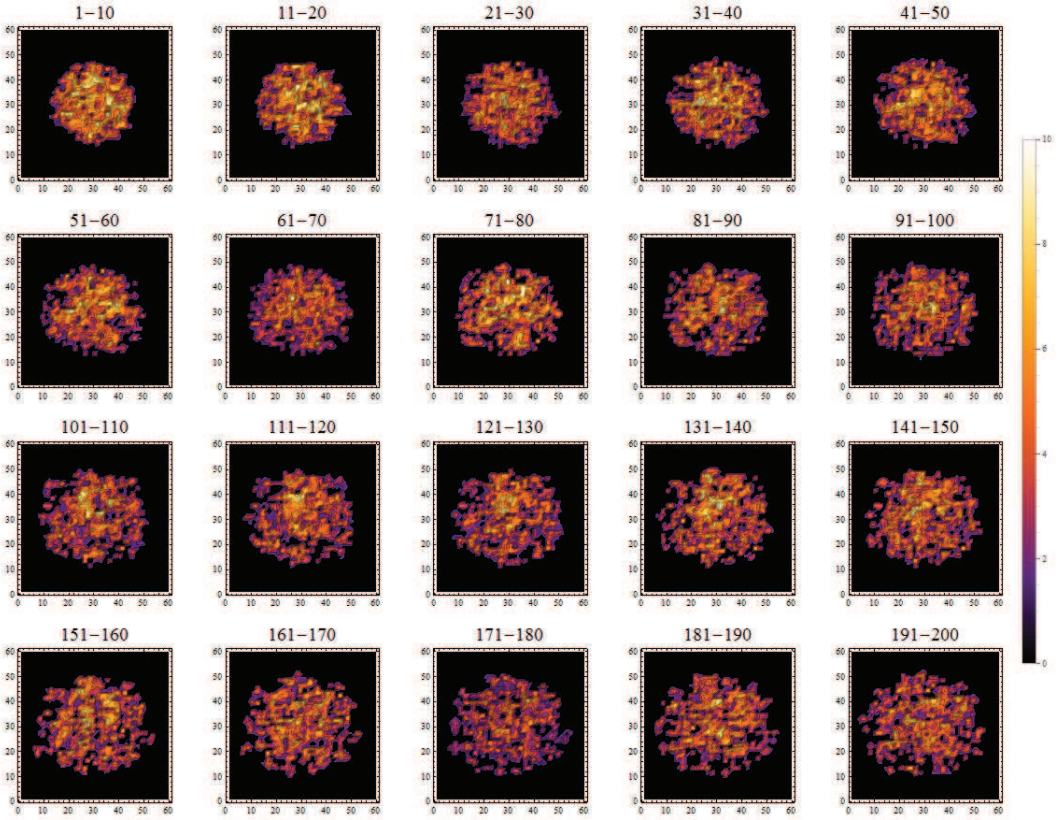
Obrázek 6.6: Ukázka průběhu pravidla z 5.skupiny



Obrázek 6.7: Ukázka průběhu pravidla z 6.skupiny

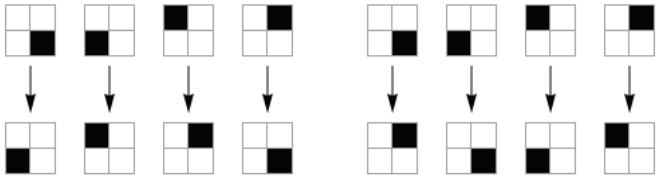


Obrázek 6.8: Ukázka průběhu pravidla ze 7.skupiny – pravidlo 19



Obrázek 6.9: Ukázka průběhu pravidla ze 7.skupiny – pravidlo 10

složek, dojdeme k překvapivému zjištění. Ukázalo se, že složka 10 představovala pravotočivý pohyb buňky a složka 19 je jejím odrazem, u ní se buňka pohybuje ve směru levotočivém. Tímto tříděním jsme získali poznatky o tom, kterým směrem soustře-



(a) pravidlo 10 (b) pravidlo 19

Obrázek 6.10: Schéma vhodných pravidel první složky

dit naše další hledání, a také se nám radikálně snížil počet pravidel, ze kterých jsme potřebovali vybrat nejvhodnější kandidáty.

Nyní jsme věděli, že hledané pravidlo či pravidla bude začínat buď číslem 10- nebo 19-, ale stále nám zbývalo najít dvě zbylé dvě složky, které by dohromady s první složkou dávaly nejvhodnější pravidlo pro simulování difúze. I takto snížená oblast hledání nám stále dávala něco okolo 34 000 možných pravidel. Ze zbylých dvou složek má složka pro dvě buňky větší vliv na tvar průběhu nežli složka pro tři buňky, a proto jsme se v našem dalším zkoumání zaměřili právě na tuto složku. Postupovali jsme tak, že jsme opět vygenerovali průběh všech možných pravidel pro blok se dvěma buňkami, kterých je přesně 720, a udělali jsme tak pro několik vybraných hodnot ze třetí složky. Toto vyhodnocování jsme udělali jak pro pravidla, která začínají číslem 10, tak i pro ta, která mají na začátku číslo 19. Matice, se kterou jsme tento průzkum realizovali, měla rozměry 100×100 a evoluce probíhala po dobu 1000 kroků. Porovnávali jsme výsledné ContourPloty, pro úsek dlouhý 10 evoluční kroků v době mezi kroky 990 a 1000. Získané průběhy jsme opět analyzovali vizuálně, jelikož vynesení do grafu by nám nedokázalo poskytnout dostatečně podrobnou informaci o směru a tvaru šíření částic tak, jak jim to přikazuje dané pravidlo.

Takto jsme získali pravidla, která se jevila jako nejlepší kandidáti pro simulování difúzního pohybu.

7. Ohraničený prostor

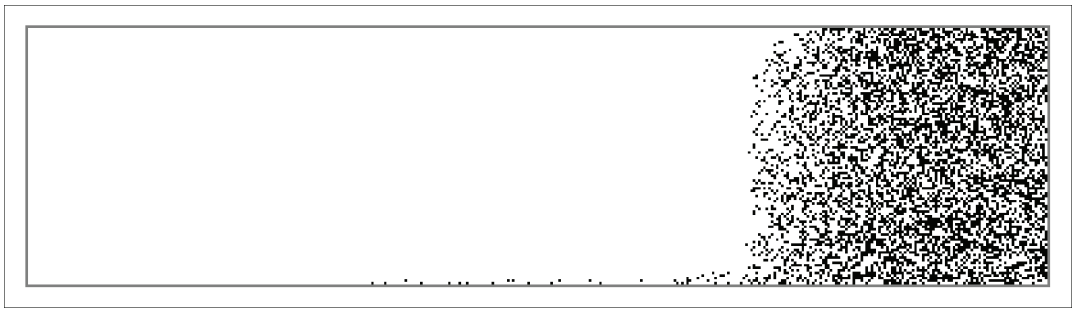
Dosud jsme používali matici, jejíž hranice byly vzájemně propojeny a vznikl nám tak toroidní systém. Takto definované prostředí nám ale nenabízí příliš možností pro modifikace a úpravy. Abychom tak mohli udělat, potřebujeme mít způsob jak vytvořit hranice a překážky pro pohybující se částice v našem systému.

Udělali jsme to tedy tak, že v matici jsme *překážku*, které jsme přiřadili hodnotu, která je odlišná od hodnot pro plnou a prázdnou buňku. Jelikož jsme v našem případě přiřadili prázdné buňce hodnotu 0 a plné hodnotu 1, *překážku* jsme definovali hodnotou 2. Jelikož se evoluční pravidla týkají pouze bloku a ne i sousedů, výskyt nové neznámé v matici nemá na naše pravidla jakýkoliv vliv. Pro evoluci bloku, který obsahuje buňky s *překážkou* jsme stanovili, že tento blok zůstane nezměněn. Každý blok má 4 buňky a v případě, že systém bude ohraničen, se bloky, obsahující buňky *překážky*, budou vyskytovat pouze ve dvou variantách: blok bude mít 2 buňky *překážky*, nebo blok bude mít 3 buňky *překážky*. V bloku se 3 buňkami *překážky*, což mohou být pouze rohové bloky, se nenabízí žádný volný prostor pro pohyb částice, tudíž je zbytečné aplikovat na něj nějaké pravidlo pro evoluci, a blok se 2 buňkami *překážky* má prostor pro pohyb pouze jedné částice a to pouze posun ve vertikálním, nebo horizontálním směru. Rozhodli jsme se, že i v tomto případě neaplikujeme pravidlo, které by s částicí pohybovalo, jelikož by to nemělo nijak výrazný vliv na celkový průběh evoluce.

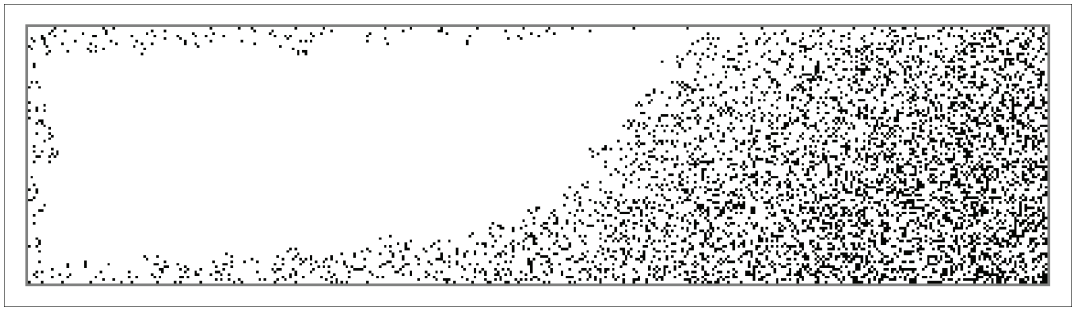
Ze začátku jsme evoluci ohraničeného systému realizovali tak, že jsme při lichém kroku rozkládali na bloky neohraničenou matici a po aplikaci pravidel jsme výslednou matici opět ohraničili. V sudém kroku jsme pracovali s ohraničenou maticí. K tomu se dala použít pouze verze generátoru s cyklem Do. Později jsme ale zjistili, že funkce RotateRight si bez problémů poradí i se systémem, který je ohraničený. Proto jsme od té chvíle používali uspořádání pro generování lichých a sudých kroků tak, jak je uvedeno v podkapitole 4.3.

Narazili jsme zde však na problém, který vznikl při kontaktu částic s bariérou.

Vlivem definice první složky pravidla, která představovala pravotočivou rotaci (složka s číslem 10) nebo levotočivou rotaci (složka s číslem 10), se částice začaly posunovat podél jedné ze stěn (viz Obr. 7.1 a Obr. 7.2).



Obrázek 7.1: Ukázka nevyhovujícího průběhu po 500 krocích



Obrázek 7.2: Ukázka nevyhovujícího průběhu po 10000 krocích

Když jsme hledali způsob, jak tento problém vyřešit, první věc, která nás napadla, bylo pro každý evoluční krok vybrat náhodně jedno pravidlo ze všech 414 700. Vycházeli jsme z toho, že skutečné částice se pohybují zcela náhodně a není možné přesně stanovit, kde se budou v daném okamžiku nacházet.

V kódu pro vytváření evolucí s náhodnou volbou pravidel pro každý krok jsme pracovali s variantou generování pomocí cyklu Do. Mechanismus aplikace pravidel a přepínání mezi lichými a sudými kroky se nemění, pouze jsme vytvořili tři nové proměnné. Nazvali jsme je x1, x2 a x3 a přiřadili vlastnost náhodného výběru. Toho lze v programu *Mathematica* dosáhnout tak, že místo klasického =, použijeme :=. Díky tomu proměnné neobsahovaly pouze jednu náhodně vybranou hodnotu, kterou by si zachovaly po celou dobu výpočtu, ale místo toho nyní obsahovaly samotnou funkci RandomChoice, tak jak jsme ji pro každou proměnnou nadefinovali. Ve výsledku jsme tedy vždy při zavolání těchto proměnných dostali hodnoty, které se při každém zavolání měnily. Každá z těchto tří proměnných představovala jednu ze tří složek, ze kterých

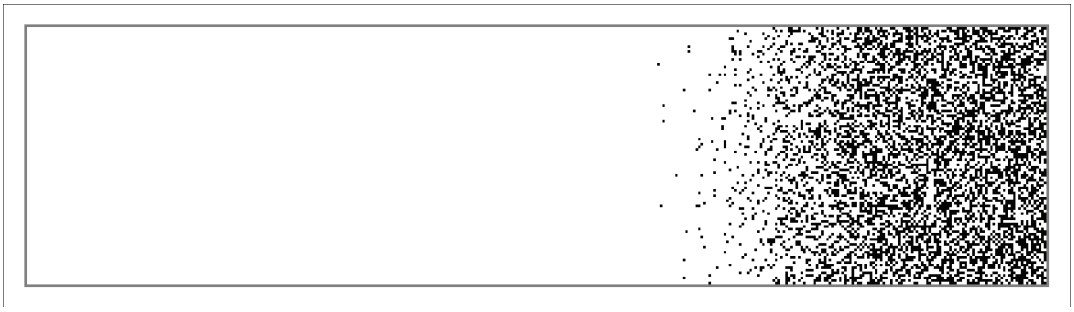
se jednotlivá pravidla skládají. Tudíž první proměnná vybírala ze 24 možných hodnot, proměnná v pořadí druhá vybírala ze 720 možných hodnot a poslední proměnná mohla vybírat opět ze 24 hodnot.

Kód pro generování takovéhoto BCA průběhu vypadá následovně:

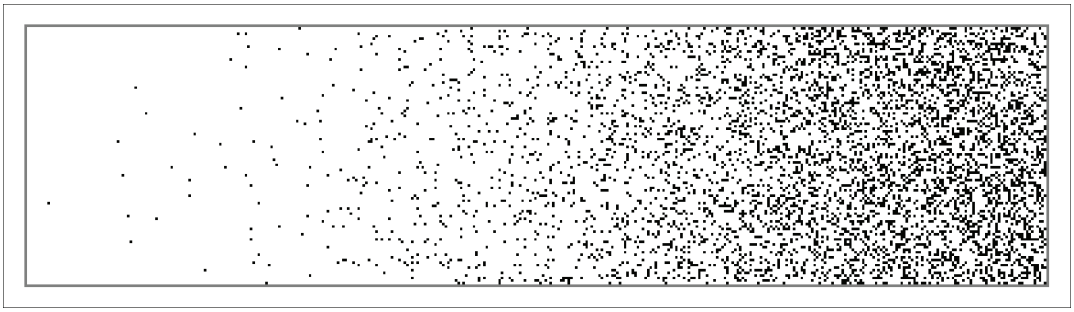
```
x1 := RandomInteger[{1, 24}]; x2 := RandomInteger[{1, 720}];
x3 := RandomInteger[{1, 24}];

matx = ArrayPad[initTube, 1, 2];
n = True;

Do[
  If[n == True,
   n = ! n;
   matx = Flatten[Partition[matx, {2, 2}] /.
       SetRule[{oneCPerm[[x1]], twoCPerm[[x2]], threeCPerm[[x3]]}],
     {{1, 3}, {2, 4}}],
   n = ! n;
   matx = RotateLeft[
     Flatten[Partition[RotateRight[matx, {1, 1}], {2, 2}] /.
       SetRule[{oneCPerm[[x1]], twoCPerm[[x2]], threeCPerm[[x3]]}],
      {{1, 3}, {2, 4}}], {1, 1}]],
 {1000}];
```

Takovýto průběh vykazoval již viditelné znaky difúzního pohybu (viz Obr. 7.3 a Obr. 7.4). Provedli jsme 10 měření pro stejnou počáteční matici, protože jsme chtěli zjistit, jak velký dopad na průběh má skutečnost, že vybíráme náhodně pravidla při každém kroku. Ukázalo se, že mezi 10-ti námi naměřenými průběhy, nedocházelo k výrazným odchylkám v průběhu. Vysvětlujeme si to tak, že každé pravidlo ovlivňovalo částice po tak krátkou dobu (pouze jeden evoluční krok), že se v pohybu částic nedokázal projevit vzor pohybu charakteristický pro dané pravidlo.

Ačkoliv jsme byli schopni simulovat difúzní pohyb náhodným výběrem pravidel,



Obrázek 7.3: Ukázka průběhu při náhodném výběru pravidel po 500 krocích



Obrázek 7.4: Ukázka průběhu při náhodném výběru pravidel po 10000 krocích

našim cílem bylo najít nějaká pevně daná pravidla, která by tento pohyb dokázala simulovat taktéž.

Zaměřili jsme se proto blíže na jednotlivé složky, ze kterých se naše pravidla skládají. K rozptýlení částic z počátečního uspořádání dochází tak, že se postupně oddělují jednotlivé částice na okrajích a rozptylují se do okolí. Z toho plyne, že největší, možná i jediný, vliv na způsob ‚jakým dojde k rozptýlení do okolí, má první složka, která obsahuje pravidla pro bloky právě s jednou částicí. Už víme, že při použití pravidel 10 a 19 dochází k výraznému pohybu částic podél stěn a zbylá pravidla naopak nejsou schopna produkovat rovnoměrný difúzní pohyb. Naší myšlenkou bylo, že pokud žádné pravidlo první složky osamoceně nedokáže simulovat difúzi v uzavřeném prostoru, mohli bychom vytvořit kombinaci těchto pravidel, kdy by se tato pravidla svým charakterem vzájemně doplňovala a tudíž bychom byli schopni vyrušit nechtěný jev cestování částic po okrajích. Už víme, že pravidla 10 a 19 z první složky představují kruhový pohyb a z kapitoly 6 jsme získali přibližnou představu o vývoji počáteční distribuce částic ve volném prostoru, pokud ji necháme vyvíjet podle jednoho z možných 24 pravidel první složky.

Rozhodli jsme se pracovat pouze s kombinacemi pravidel 10, 19 a 17. Vycházeli jsme z toho, že pravidla 10 a 19 jsou pro náš průběh stěžejní, jelikož mají na svědomí rovnoměrné rozptylování částic. Pravidlo 17 jsme použili, protože jako jediné ze zbylých pravidel první složky způsobuje rapidní pohyb do čtyř směrů, a tudíž by mělo zabránit částicím, aby se následkem použití pravidel 10 či 19 posouvaly podél stěn *překážky*. Ostatní pravidla vytvářejí průběh, který je příliš nerovnoměrný, než abychom ho mohli použít k simulování rovnoměrného difúzního průběhu.

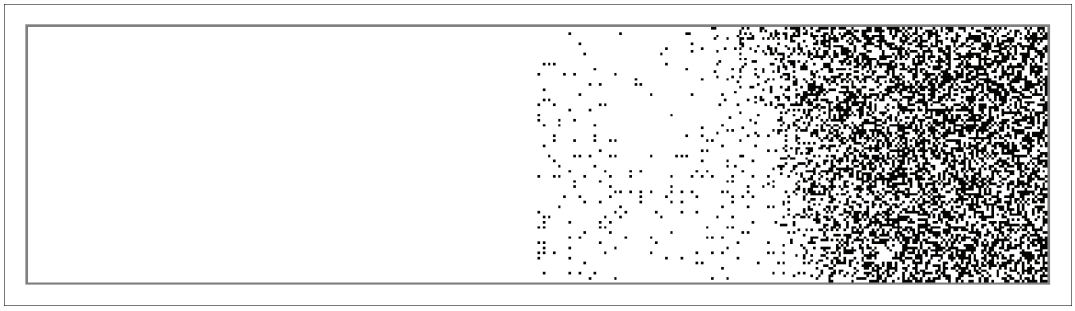
Kód, který nám umožnil generovat pravidla, která obsahují kombinace pravidel první složky, byl v základě totožný s kódem pro generování průběhů pomocí cyklu

Do. Jediná změna se týkala proměnné oneCPerm, která obsahuje transformační instrukce pro evoluci případu, kdy je v bloku jen jedna částice. Udělali jsme to tak, že jsme si vytvořili proměnnou, kterou jsme nazvali byakka a tato proměnná obsahuje uspořádanou n-tici jednotlivých pravidel, ze kterých chceme vytvořit danou kombinaci. Takto můžeme vytvářet kombinace o libovolné délce a jednotlivá pravidla můžeme prohazovat v pořadí, v jakém právě potřebujeme. Zároveň s touto proměnnou jsme vytvořili další proměnnou s názvem sts, jejíž hodnota vždy naroste o jedničku při každém evolučním kroku. Když tyto dvě proměnné dáme dohromady, výsledný tvar pro výběr pravidla pro jednu částici vypadá takto: oneCPerm [[byakka [[1 + sts - Length[byakka]*Floor[sts/Length[byakka]]]]. Dvojité hranaté závorky představují symbolický tvar funkce Part. Toto uspořádání funguje tak, že při každém kroku vydělíme hodnotu proměnné sts počtem prvků, které jsou obsaženy v proměnné byakka a zaokrouhlením této hodnoty dolů dostaneme dostaneme celé číslo, kterým opět vynásobíme počet prvků v proměnné byakka. Tak získáváme celočíselné násobky počtu proměnných, které odečteme od hodnoty konkrétního, právě probíhajícího, kroku a získáme informaci o tom, který konkrétní prvek z proměnné byakka se má v daném okamžiku použít.

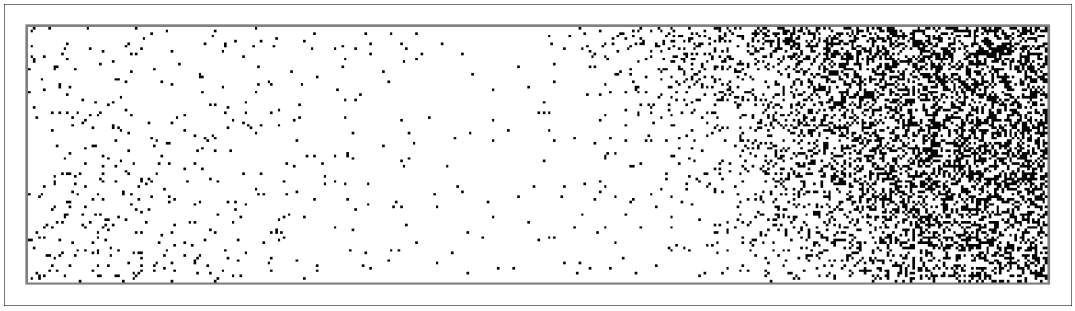
Pracovali jsme tedy s kombinacemi pravidel 10 a 19, 10 a 17 a v poslední řadě 19 a 17. Měnili jsme různě délky periody a doby, po které jednotlivá pravidla působila na daný systém a přitom jsme sledovali průběh jejich evoluce pro případ, kdy jsou částice umístěny v trubici a pro případ, kdy jsou částice na počátku koncentrovány v kruhové distribuci uprostřed volného prostoru. Takto jsme byli schopni pozorovat, jakým způsobem se částice rozptylují do prostoru. Postupovali jsme tak, že jsme různě měnili periodu, po kterou budou jednotlivá pravidla střídavě ovlivňovat evoluci dané matice částic a také jsme měnili sílu vlivu těchto pravidel tak, že jsme je nechávali působit nerovnoměrně dlouhou dobu.

Ukázalo se, že při použití kombinace pravidel první složky 10 a 19 nejsme schopni zaznamenat průběh, který by byl pro simulování difúze vhodný. Měnili jsme dobu po kterou působilo pravidlo 10 a pravidlo 19 první složky na danou matici buněk tak, že jsme začali s pravidlem 10, které působilo vždy jen po jeden evoluční krok, následované pravidlem 19, které jsme naopak nechali působit po 9 kroků. Postupně jsme zvyšovali počet kroků, po které bude pravidlo 10 působit na danou matici buněk, a zároveň jsme úměrně k tomuto navyšování zmenšovali počet kroků, po které bude pravidlo 19 působit. To jsme prováděli tak často, dokud jsme nedostali kombinaci, kdy

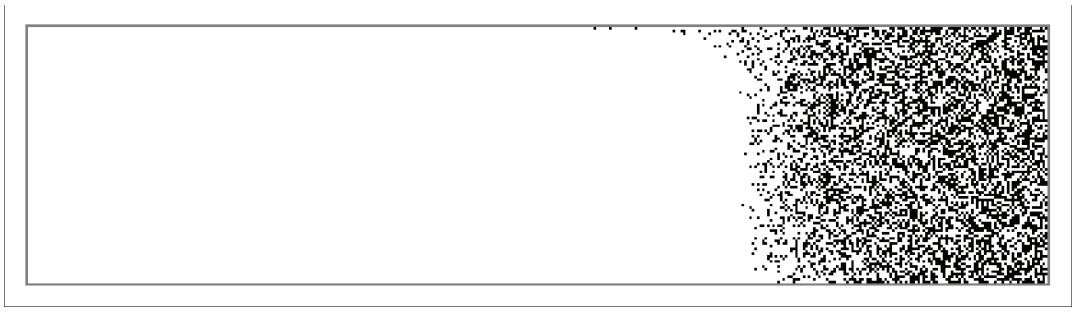
pravidlo 10 bude působit po 9 kroků a pravidlo 19 bude působit vždy pouze po jeden krok. Jako první jsme tyto kombinace nechali vyvíjet v prostředí trubice. Měli jsme matici buněk, která měla rozměry 100×200, a částice jsme měli na počátku koncentrovány v prostoru o rozměrech 100×, který zabíral celou levou polovinu trubice. Tato trubice byla samozřejmě ohraničená. Byli jsme schopni pozorovat průběhy, které jsou zobrazeny na snímcích Obr. 7.5 až Obr. 7.22.



Obrázek 7.5: Pravidlo 10 pro 1 krok a pravidlo 19 pro 9 kroků, 500 kroků

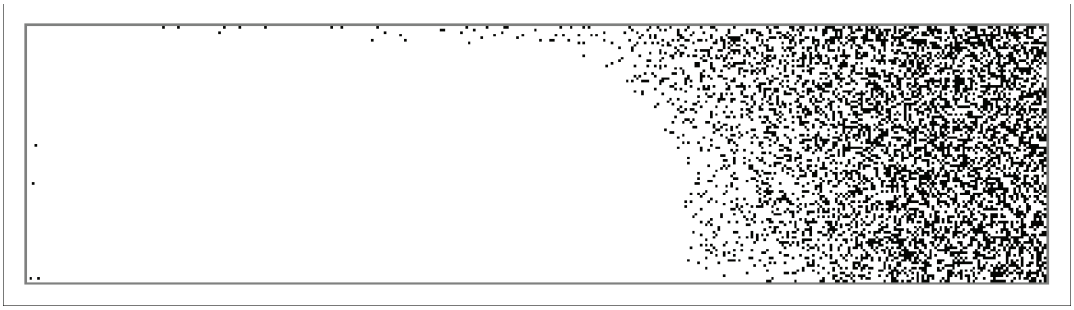


Obrázek 7.6: Pravidlo 10 pro 1 krok a pravidlo 19 pro 9 kroků, 2500 kroků

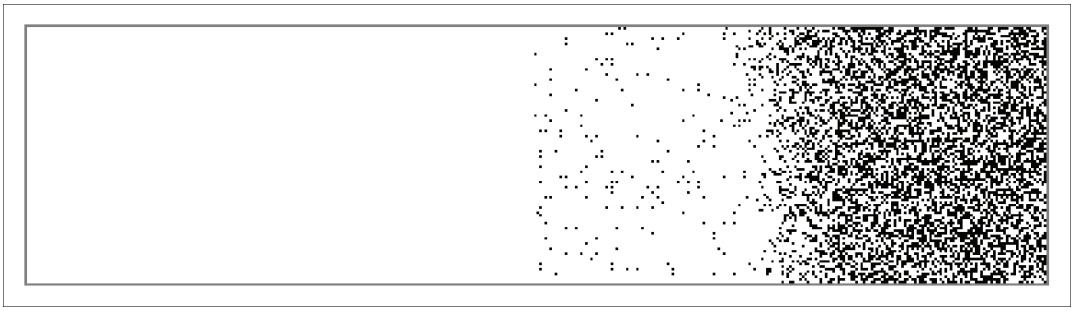


Obrázek 7.7: Pravidlo 10 pro 2 kroky a pravidlo 19 pro 8 kroků, 500 kroků

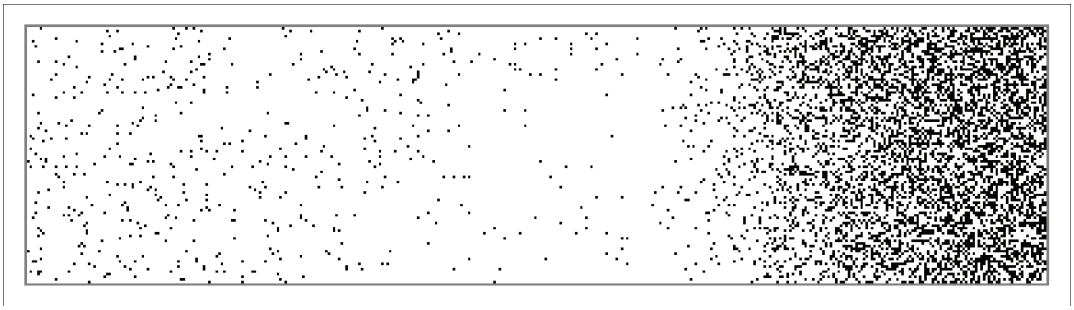
Z jednotlivých průběhů je patrné, že případy, kdy se částice pohybovaly pospolitě a rovnoměrně, místo toho aby putovaly podél stěn, nastávaly pouze když pravidla 10 a 19 působila po lichý počet kroků, nezávisle na tom, které z nich působilo delší dobu. Vyvstala zde tedy možnost, že tyto kombinace by se daly použít pro simulování



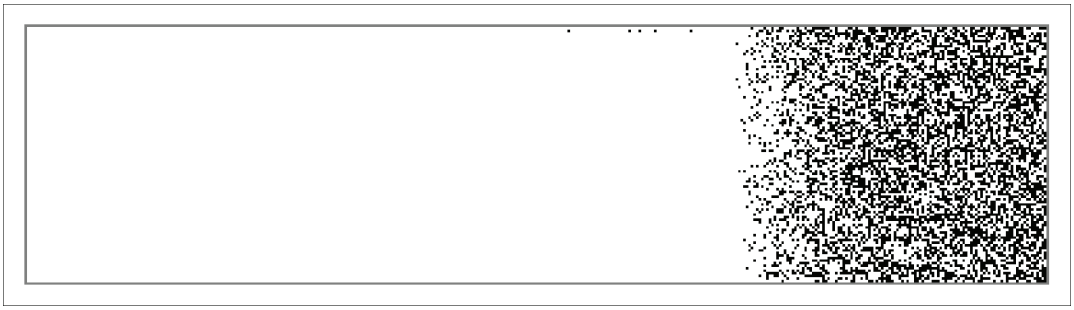
Obrázek 7.8: Pravidlo 10 pro 2 kroky a pravidlo 19 pro 8 kroků, 2500 kroků



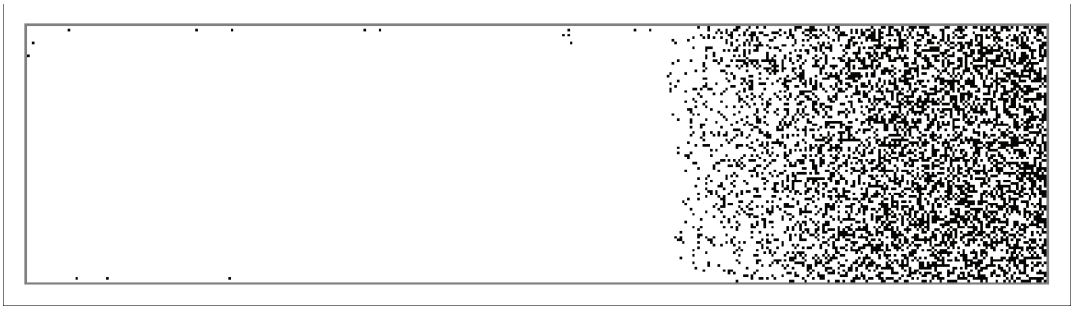
Obrázek 7.9: Pravidlo 10 pro 3 kroky a pravidlo 19 pro 7 kroků, 500 kroků



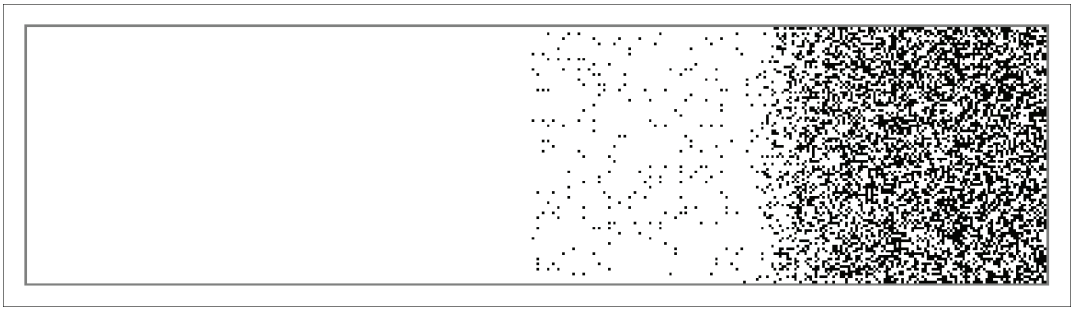
Obrázek 7.10: Pravidlo 10 pro 3 kroky a pravidlo 19 pro 7 kroků, 2500 kroků



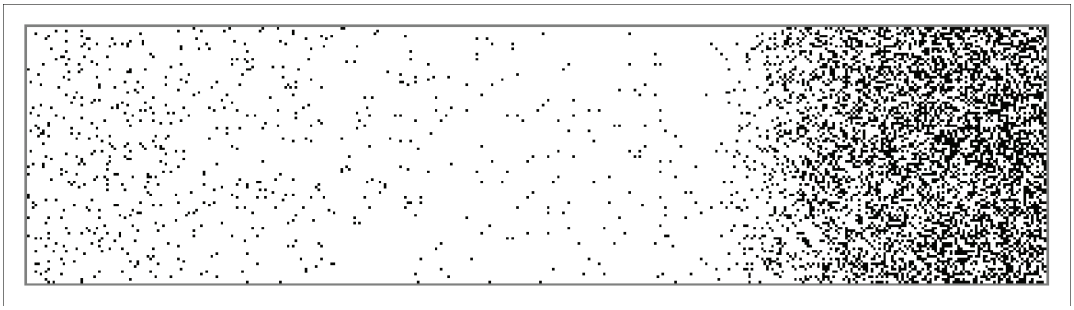
Obrázek 7.11: Pravidlo 10 pro 4 kroky a pravidlo 19 pro 6 kroků, 500 kroků



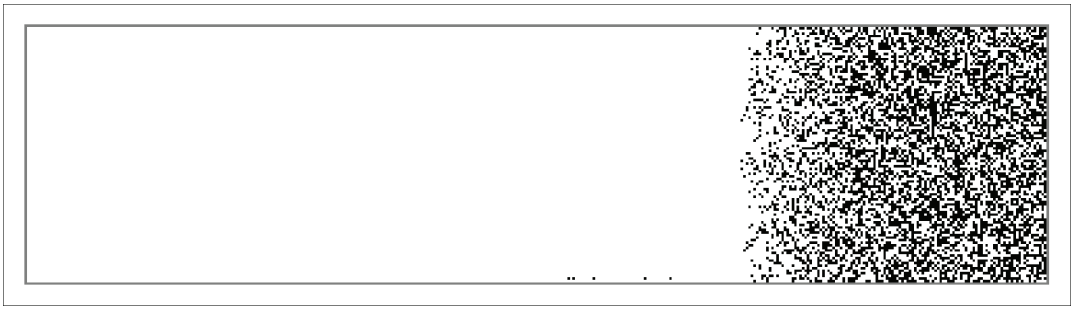
Obrázek 7.12: Pravidlo 10 pro 4 kroky a pravidlo 19 pro 6 kroků, 2500 kroků



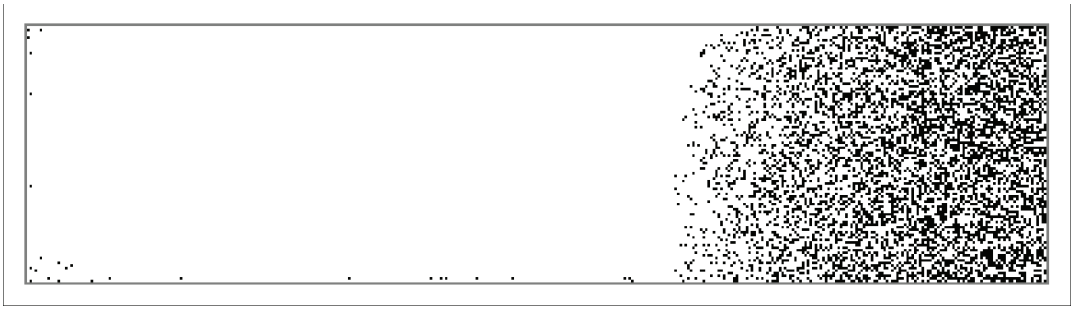
Obrázek 7.13: Pravidlo 10 pro 5 kroků a pravidlo 19 pro 5 kroků, 500 kroků



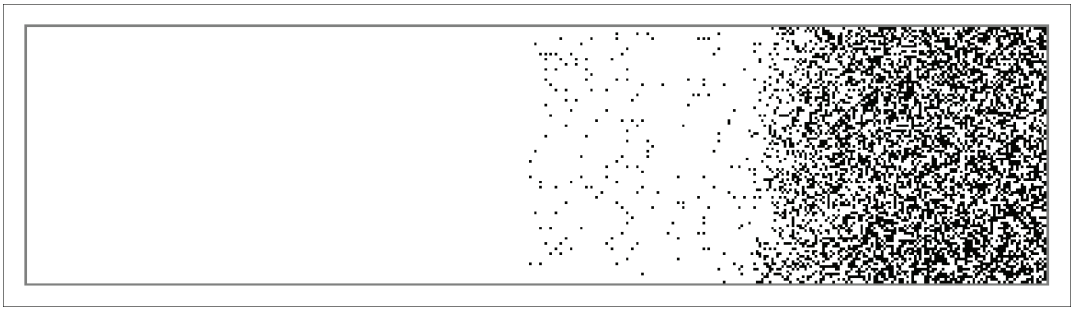
Obrázek 7.14: Pravidlo 10 pro 5 kroků a pravidlo 19 pro 5 kroků, 2500 kroků



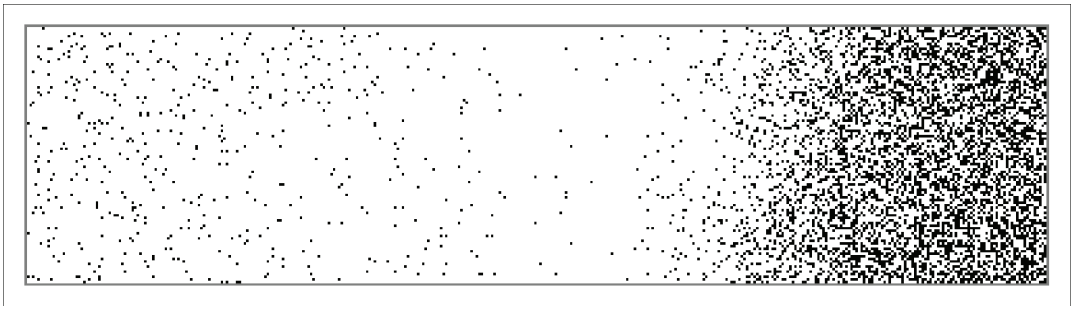
Obrázek 7.15: Pravidlo 10 pro 6 kroků a pravidlo 19 pro 4 kroky, 500 kroků



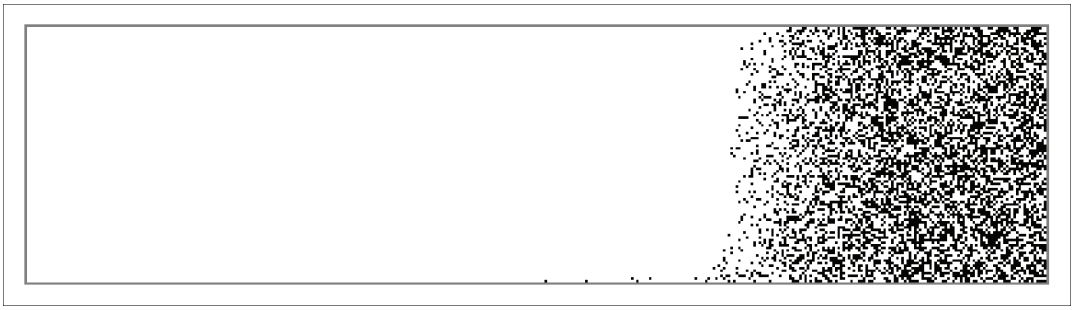
Obrázek 7.16: Pravidlo 10 pro 6 kroků a pravidlo 19 pro 4 kroky, 2500 kroků



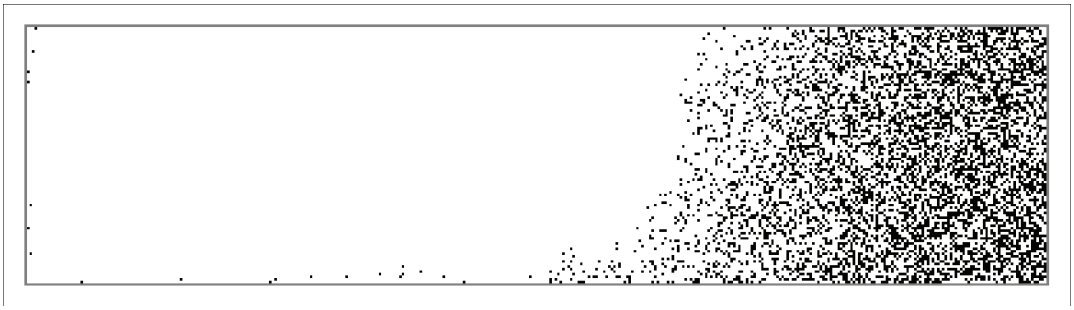
Obrázek 7.17: Pravidlo 10 pro 7 kroků a pravidlo 19 pro 3 kroky, 500 kroků



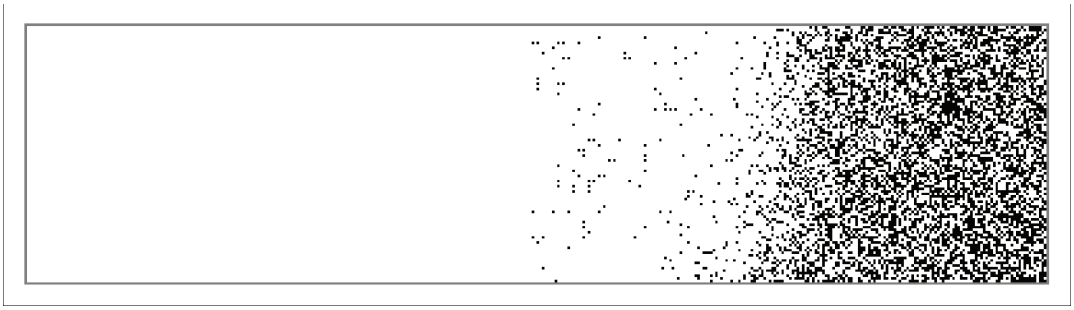
Obrázek 7.18: Pravidlo 10 pro 7 kroků a pravidlo 19 pro 3 kroky, 2500 kroků



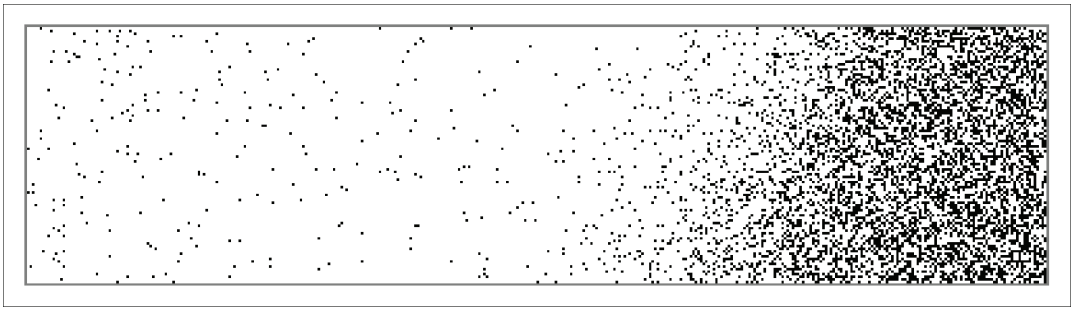
Obrázek 7.19: Pravidlo 10 pro 8 kroků a pravidlo 19 pro 2 kroky, 500 kroků



Obrázek 7.20: Pravidlo 10 pro 8 kroků a pravidlo 19 pro 2 kroky, 2500 kroků



Obrázek 7.21: Pravidlo 10 pro 9 kroků a pravidlo 19 pro 1 krok, 500 kroků



Obrázek 7.22: Pravidlo 10 pro 9 kroků a pravidlo 19 pro 1 krok, 2500 kroků

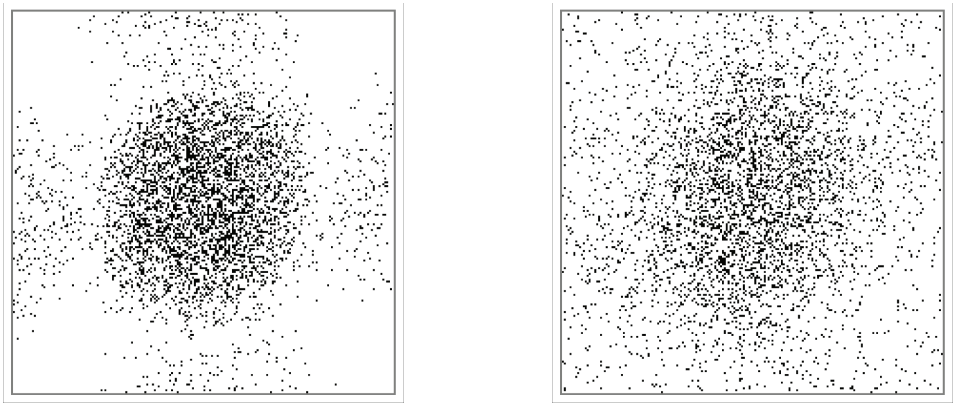
difúze. Abychom si to mohli potvrdit, bylo ještě třeba zjistit, jak vypadá jejich průběh v případě, kdy je částicím umožněn pohyb do všech směrů, ne jen do jednoho. Použili jsme k tomu ohraničenou matici o rozměrech 200×200, která uprostřed obsahovala kruhovou distribuci částic o poloměru 50 buněk. Vývoj jsme provedli pro všechny kombinace pravidel první složky 10 a 19. Výsledné průběhy lze vidět na Obr. 7.23 až Obr. 7.31.

Zjistili jsme, proč u případů, kdy obě pravidla působí po lichý počet kroků, dochází k rovnoměrnému pohybu částic v trubici. Na Obr. 7.23, Obr. 7.25, Obr. 7.27, Obr. 7.29 a Obr. 7.31 vidíme, že v průběhu evoluce dochází k oddělování částic pouze v horizontálním a vertikálním směru, které tak svým pohybem vytváří tvar podobný kříži. Z toho plyne, že ačkoliv lze tuto kombinaci pravidel použít při simulování difúzního pohybu v trubici, jako univerzální pravidlo pro simulování difúze v rozličných uspořádáních se použít nedá.

Ještě jen zmíníme, že v případě počáteční distribuce ve tvaru kruhu, pozorujeme naopak u kombinací pravidel 10 a 19, které obě působí po sudý počet kroků, rovnoměrný pohyb částic do všech směrů.

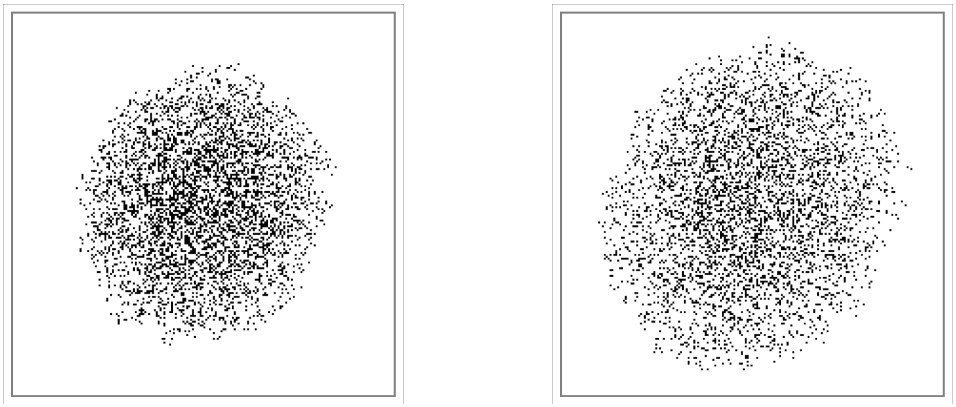
Tato skutečnost nás utvrdila v tom, že naše hledané pravidlo získáme pouze vytvořením kombinace 10 a 17 nebo kombinace 19 a 17. Definice pravidel pro jednu buňku 10 a 19 se od sebe liší pouze ve směru, ve kterém dochází k rotaci buněk. Tudíž by se průběhy u pravidel s kombinací 10 a 17 a kombinací 19 a 17 neměly výrazně od sebe lišit.

Rozhodli jsme se tedy, že se zaměříme pouze na kombinaci pravidel první složky 10



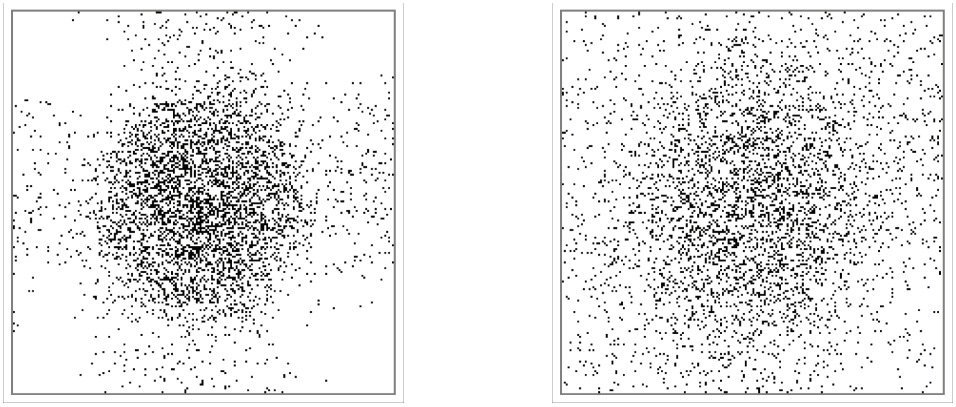
(a) 500 kroků (b) 1500 kroků

Obrázek 7.23: Pravidlo 10 pro 1 krok a pravidlo 19 pro 9 kroků



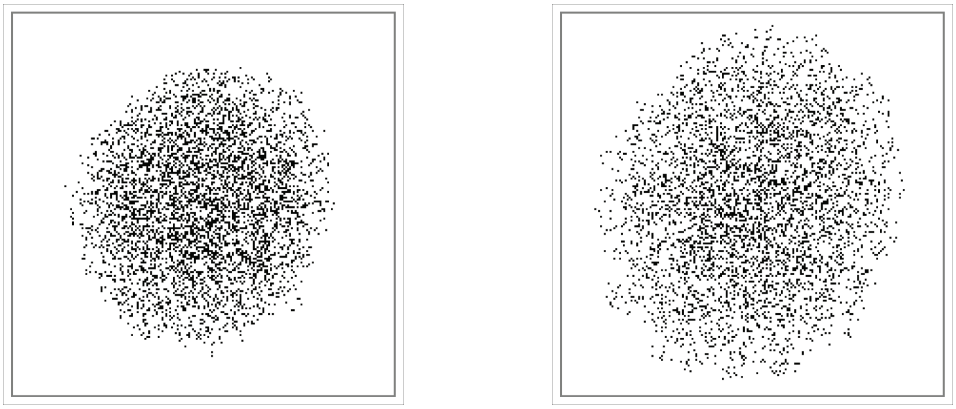
(a) 500 kroků (b) 1500 kroků

Obrázek 7.24: Pravidlo 10 pro 2 kroky a pravidlo 19 pro 8 kroků



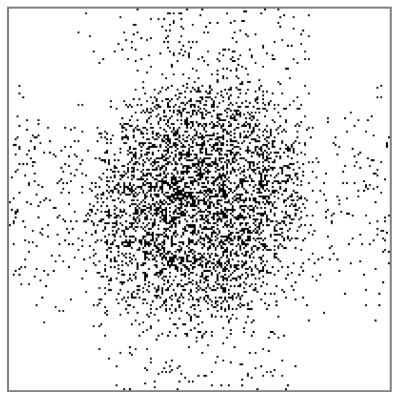
(a) 500 kroků (b) 1500 kroků

Obrázek 7.25: Pravidlo 10 pro 3 kroky a pravidlo 19 pro 7 kroků

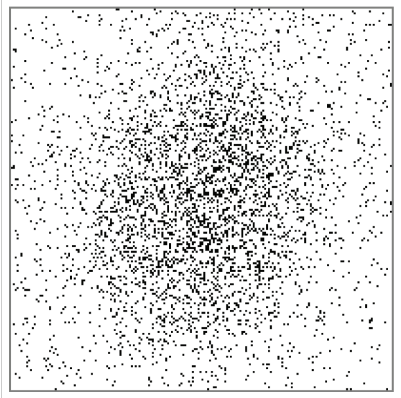


(a) 500 kroků (b) 1500 kroků

Obrázek 7.26: Pravidlo 10 pro 4 kroky a pravidlo 19 pro 6 kroků

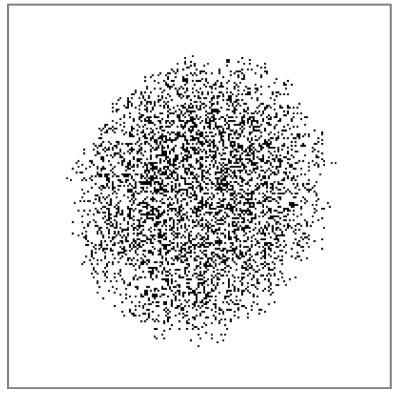


(a) 500 kroků



(b) 1500 kroků

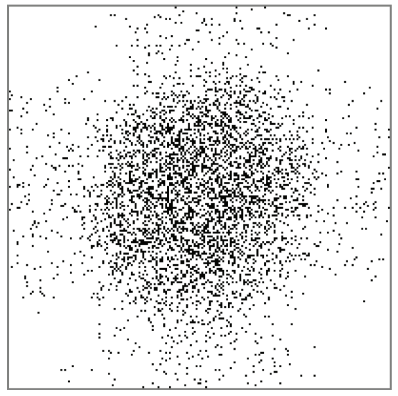
Obrázek 7.27: Pravidlo 10 pro 5 kroků a pravidlo 19 pro 5 kroků



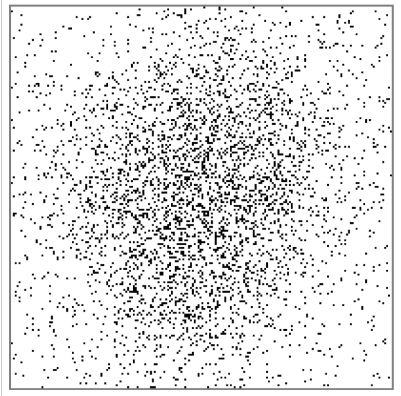
(a) 500 kroků

(b) 1500 kroků

Obrázek 7.28: Pravidlo 10 pro 6 kroků a pravidlo 19 pro 4 kroky



(a) 500 kroků

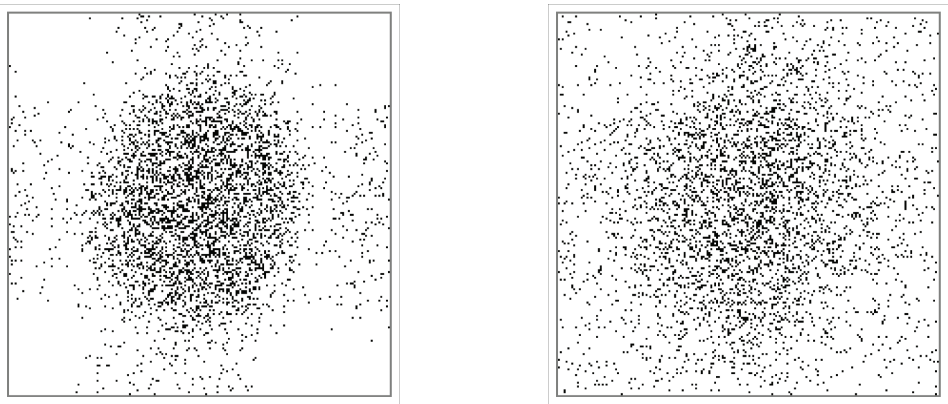


(b) 1500 kroků

Obrázek 7.29: Pravidlo 10 pro 7 kroků a pravidlo 19 pro 3 kroky

(a) 500 kroků (b) 1500 kroků

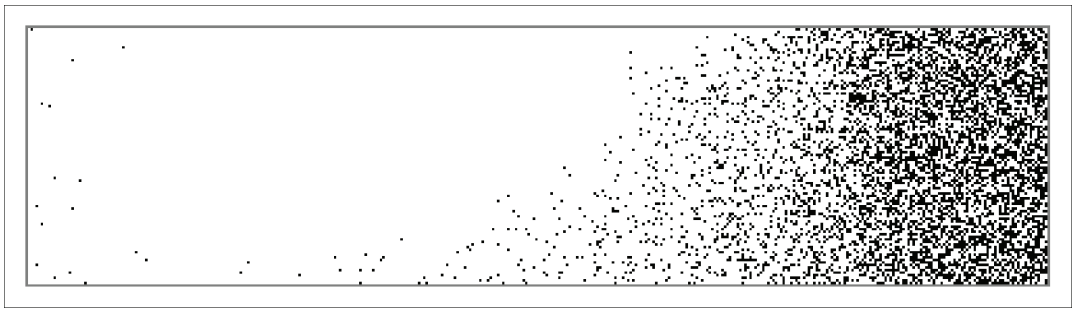
Obrázek 7.30: Pravidlo 10 pro 8 kroků a pravidlo 19 pro 2 kroky



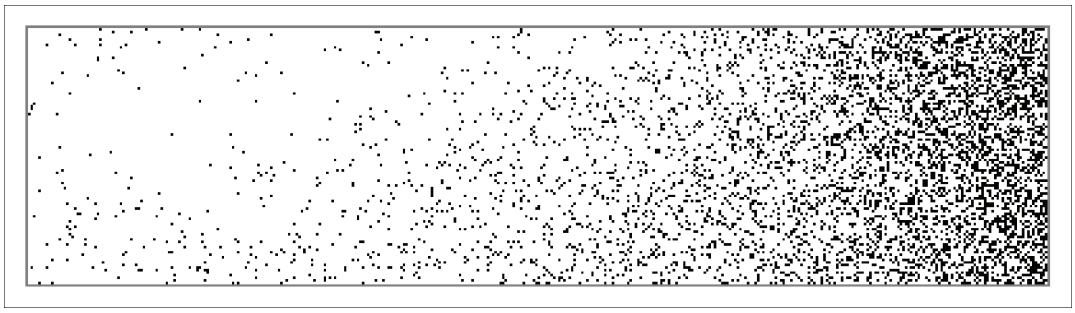
(a) 500 kroků (b) 1500 kroků

Obrázek 7.31: Pravidlo 10 pro 9 kroků a pravidlo 19 pro 1 krok

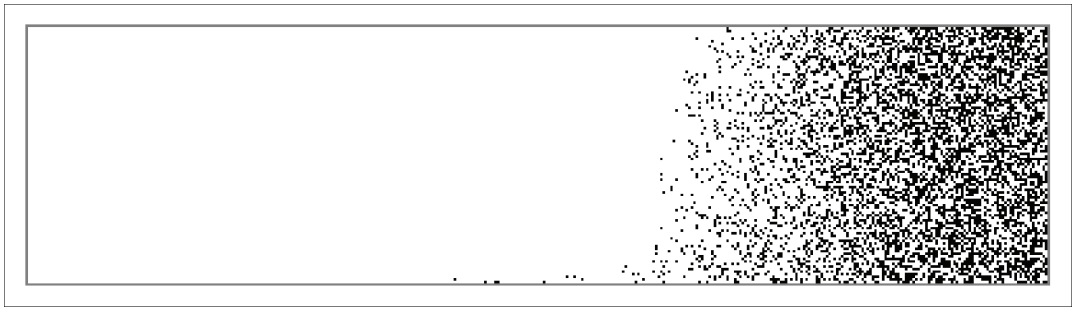
a 17. Nejdříve jsme sledovali průběh pro případy, kdy každé z obou pravidel ovlivňuje vývoj systému po rozdílnou délku kroků. Začali jsme s případem, kdy pravidlo 10 působí vždy po dobu pouze jednoho kroku a pravidlo 17 naopak působí po dobu 7 kroků. Postupně jsme zvyšovali počet kroků pro pravidlo 10 a zároveň jsme, úměrně k tomuto zvyšování, snižovali počet kroků pro pravidlo 17, až jsme skončili s kombinací délek kroků zrcadlově obrácenou k počáteční kombinaci. Částice jsme opět nechali vyvíjet v prostředí trubice. Na Obr. 7.32 až Obr. 7.45 jsou vidět výsledné průběhy. Zjistili jsme, že pouze v případě, kdy obě pravidla působí po stejně dlouhou dobu (Obr. 7.38 a Obr. 7.39), jsme schopni pozorovat rovnoměrný pohyb částic aniž by docházelo ke zvýšené aktivitě částic, které jsou ve styku s jednou ze stěn.



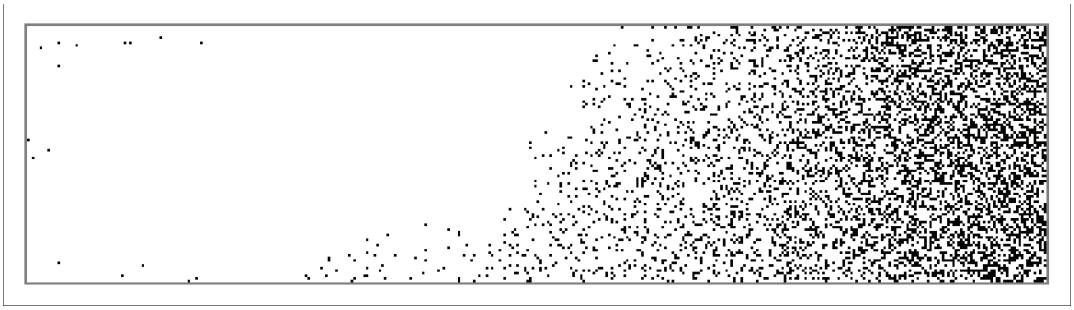
Obrázek 7.32: Pravidlo 10 pro 1 krok a pravidlo 17 pro 7 kroků, 500 kroků



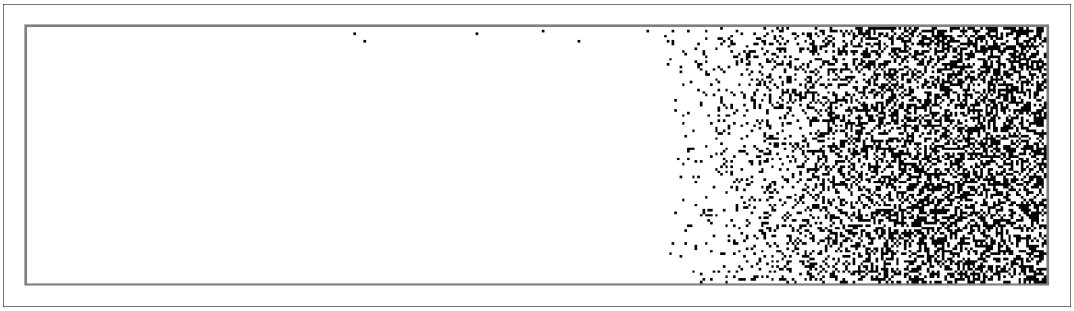
Obrázek 7.33: Pravidlo 10 pro 1 krok a pravidlo 17 pro 7 kroků, 2000 kroků



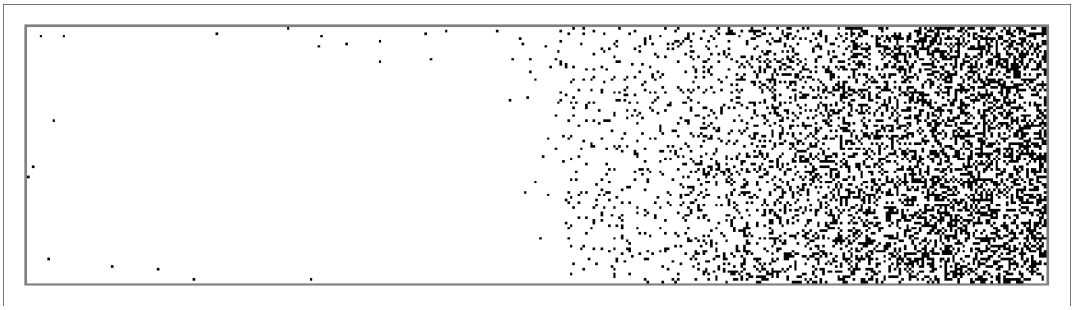
Obrázek 7.34: Pravidlo 10 pro 2 kroky a pravidlo 17 pro 6 kroků, 500 kroků



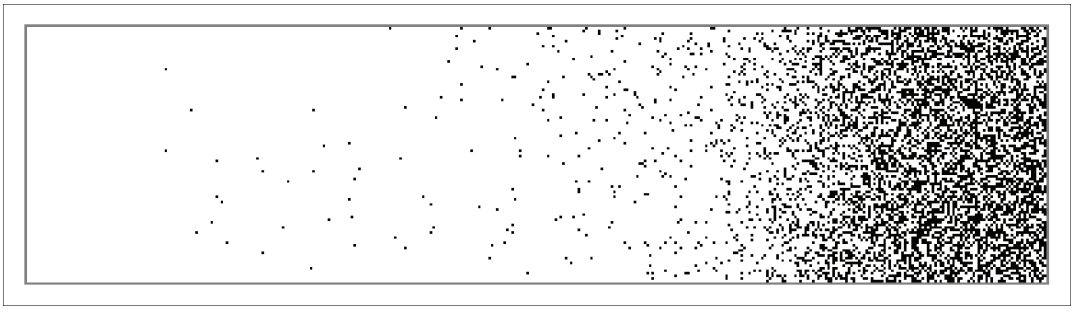
Obrázek 7.35: Pravidlo 10 pro 2 kroky a pravidlo 17 pro 6 kroků, 2500 kroků



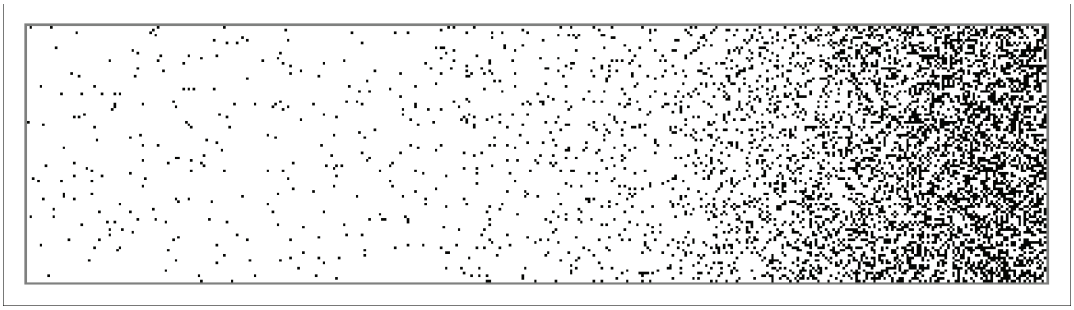
Obrázek 7.36: Pravidlo 10 pro 3 kroky a pravidlo 17 pro 5 kroků, 500 kroků



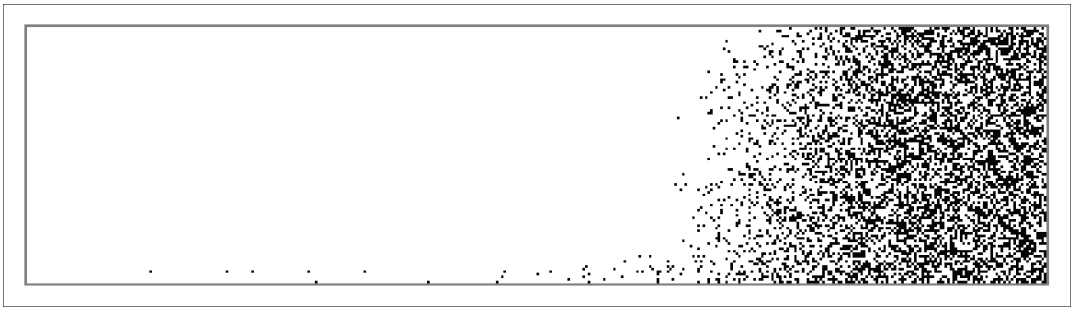
Obrázek 7.37: Pravidlo 10 pro 3 kroky a pravidlo 17 pro 5 kroků, 2500 kroků



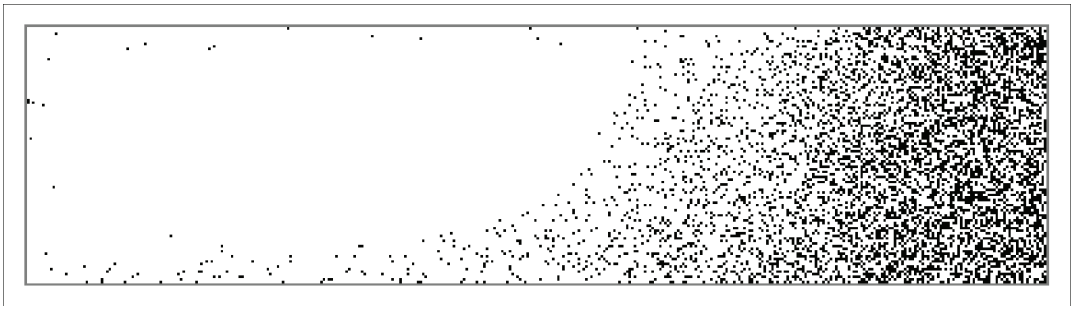
Obrázek 7.38: Pravidlo 10 pro 4 kroky a pravidlo 17 pro 4 kroky, 500 kroků



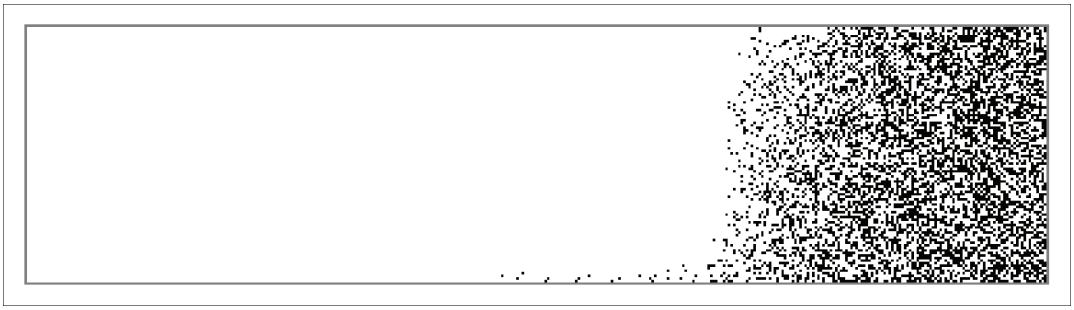
Obrázek 7.39: Pravidlo 10 pro 4 kroky a pravidlo 17 pro 4 kroky, 2500 kroků



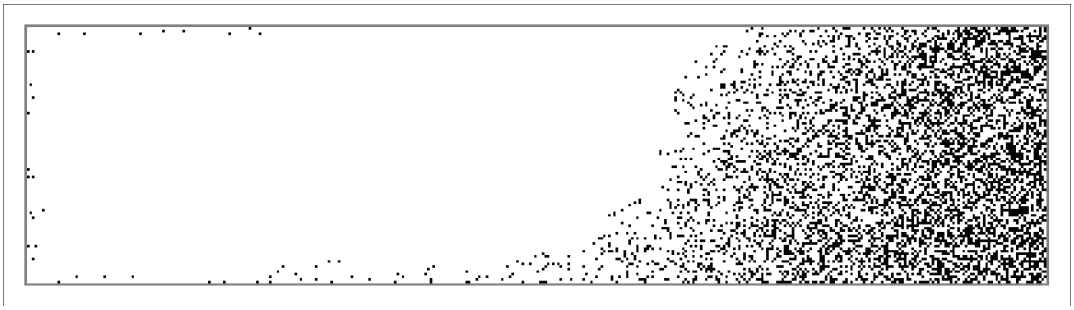
Obrázek 7.40: Pravidlo 10 pro 5 kroků a pravidlo 17 pro 3 kroky, 500 kroků



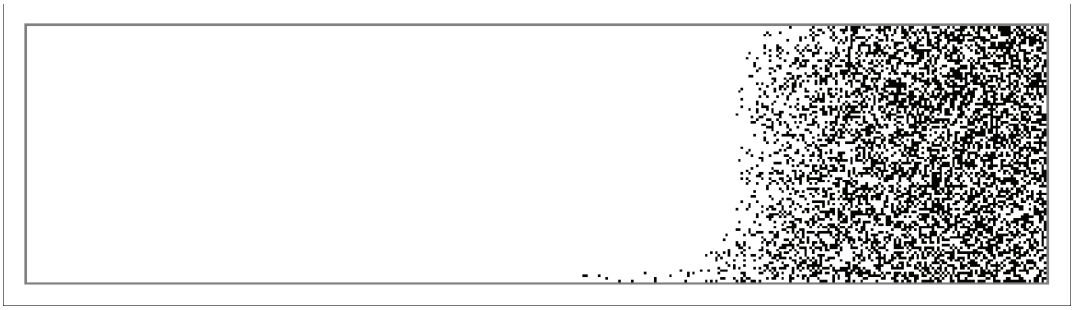
Obrázek 7.41: Pravidlo 10 pro 5 kroků a pravidlo 17 pro 3 kroky, 2500 kroků



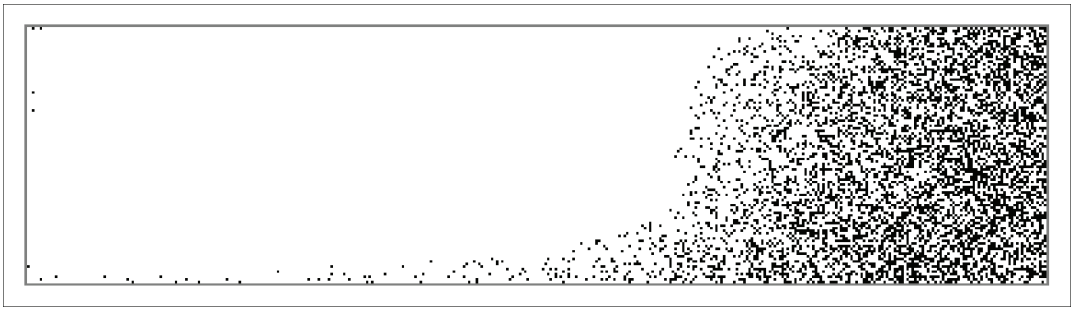
Obrázek 7.42: Pravidlo 10 pro 6 kroků a pravidlo 17 pro 2 kroky, 500 kroků



Obrázek 7.43: Pravidlo 10 pro 6 kroků a pravidlo 17 pro 2 kroky, 2500 kroků



Obrázek 7.44: Pravidlo 10 pro 7 kroků a pravidlo 17 pro 1 krok, 500 kroků



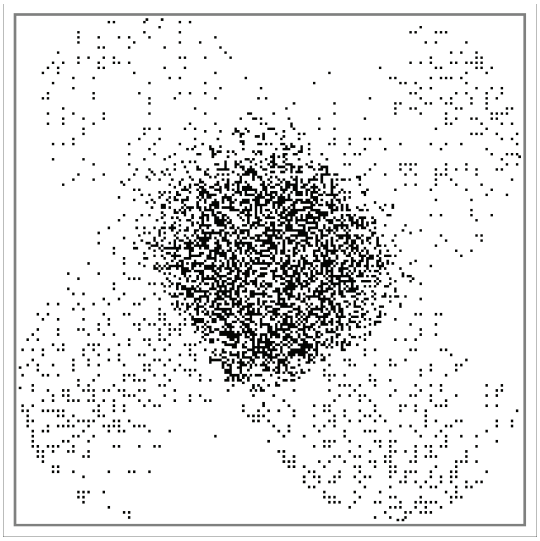
Obrázek 7.45: Pravidlo 10 pro 7 kroků a pravidlo 17 pro 1 krok, 2500 kroků

Od této chvíle jsme se věnovali jen případům, kdy obě pravidla působí po stejně dlouhý počet kroků. Na Obr. 7.38 a Obr. 7.39 je zobrazen průběh kombinace 10 a 17 pro případ kdy obě pravidla určují evoluční průběh systému vždy po dobu čtyř kroků. Položili jsme si tedy otázku, jak mohou vypadat průběhy této kombinace pravidel, pokud bude doba, po kterou budou působit na systém větší či menší nežli čtyři kroky? Začali jsme tedy s případem, kdy obě pravidla aplikujeme vždy jen po dobu jednoho kroku a postupně jsme tuto dobu zvětšovali, až do hodnoty šestnácti kroků. Sledovali jsme, jak se částice pod vlivem daných pravidel pohybují v případě, kdy jsou umístěny do trubice a v případě, kdy jsou koncentrovány do kruhu ve středu čtvercového prostoru.

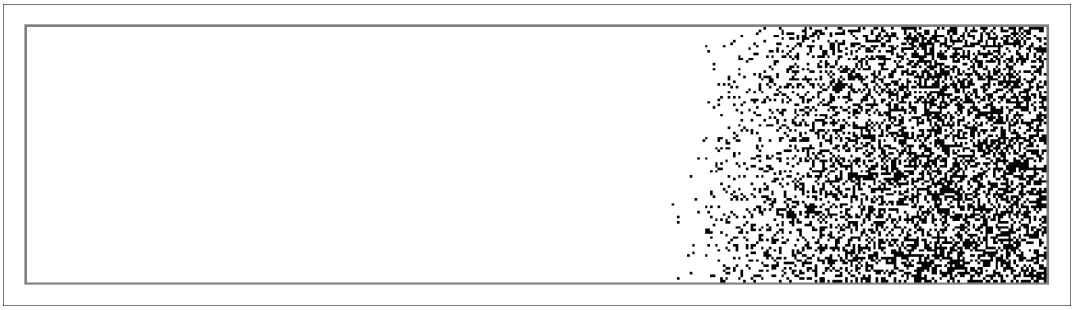
Z naměřených průběhů jsme vypozorovali, že pokud délka, po kterou pravidla jednotlivě působila na systém nenabývala hodnoty tři či se nerovnala násobkům čtyř, docházelo, v případě pohybu v trubici, k výraznému pohybu částic podél jedné ze stěn čímž se stávaly pro náš záměr zbytečnými. Tyto délky doby naopak vykazují velmi rovnoměrný rozptyl v případě, kdy začínáme s kruhovou distribucí částic. Úplně opačně je tomu právě v případech, kdy doba působení pravidel na systém je rovna celočíselnému násobku čísla čtyři. Pokud sledujeme jejich průběh v trubici, vidíme, že částice se velmi rovnoměrně a také rychle pohybují z míst s vysokou koncentrací do míst s koncentrací nižší. Velký problém nastává v případě, kdy jsou částice na počátku koncentrovány v kruhu. Dochází zde totiž k rychlému pohybu částic do všech čtyř rohů (Obr. 7.46), což si vysvětlujeme tím, že v těchto případech má pravidlo 17 příliš velký vliv na pohyb částic, a to se právě odráží ve směru pohybu částic.

Nakonec nám tedy zůstal případ, kdy obě pravidla působí střídavě po dobu tří kroků. Když se blíže podíváme na průběh této kombinace, pro případ trubice (Obr. 7.47 až Obr. 7.50) a kruhové distribuce (Obr. 7.51 a Obr. 7.54), vidíme, že právě tato kombinace vyhovuje oběma prostředím a tudíž jsme, po dlouhém pátrání, získali pravidlo, které splňuje všechny požadavky nutné k tomu, abychom ho mohli použít pro simulování difúze. Samozřejmě i u této kombinace se nemůžeme vyvarovat případů, kdy se při probíhající evoluci od hlavní masy částic oddělí malé množství. Tento jev je však pro tuto konkrétní kombinaci, v porovnání s ostatními dosud zkoumanými, minimální.

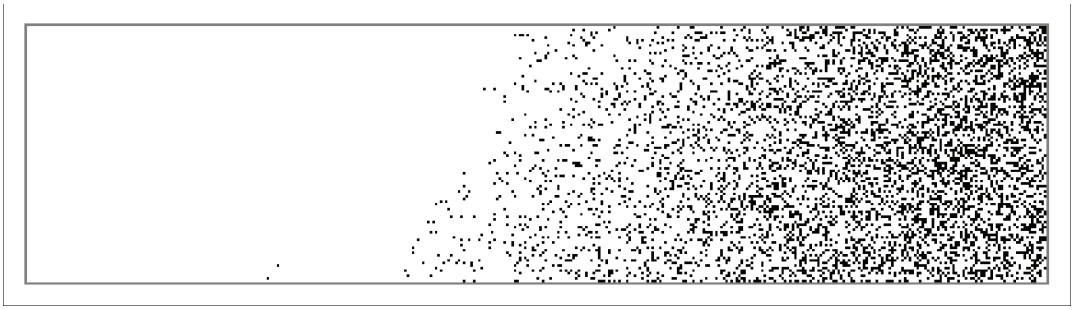
Jelikož jsme již vyřešili problém s pravidlem první složky, zbývalo nám jen vybrat vhodná pravidla pro druhou a třetí složku. Pravidlo druhé složky hraje hlavně velkou



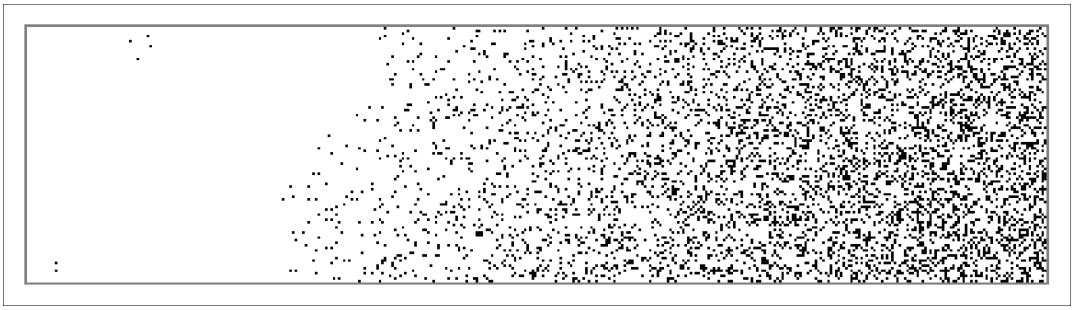
Obrázek 7.46: Kombinace pravidel 10 a 17 působící po 4 kroky; 500 kroků



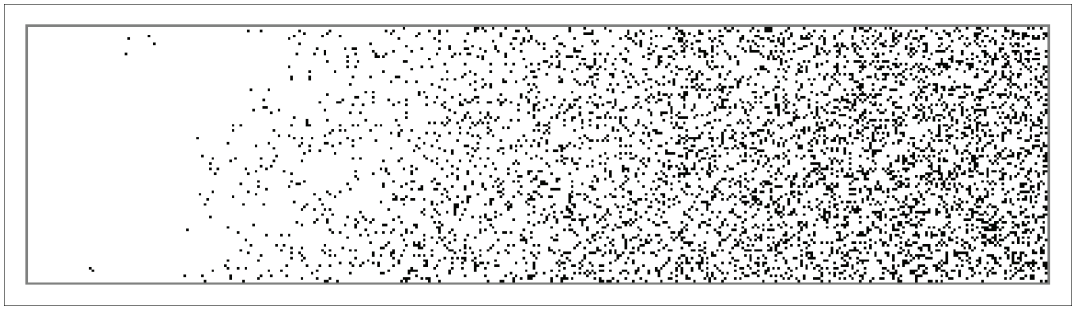
Obrázek 7.47: Kombinace pravidel 10 a 17 působící po 3 kroky; 500 kroků



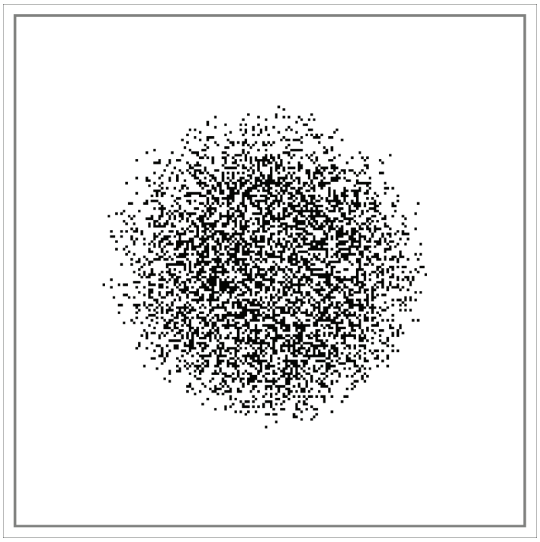
Obrázek 7.48: Kombinace pravidel 10 a 17 působící po 3 kroky; 5000 kroků



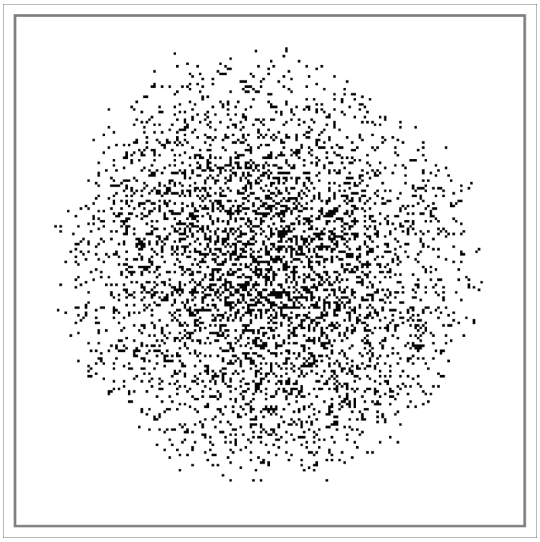
Obrázek 7.49: Kombinace pravidel 10 a 17 působící po 3 kroky; 10000 kroků



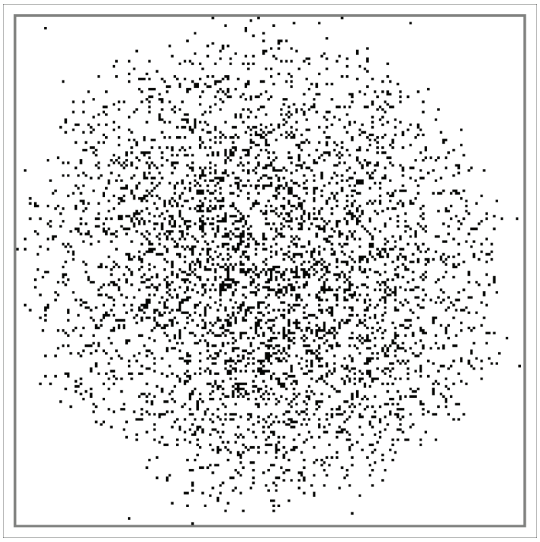
Obrázek 7.50: Kombinace pravidel 10 a 17 působící po 4 kroky; 15000 kroků



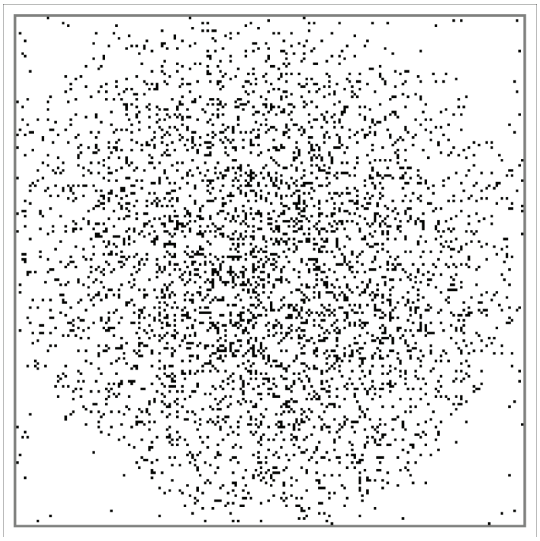
Obrázek 7.51: Kombinace pravidel 10 a 17 působící po 3 kroky; 100 kroků



Obrázek 7.52: Kombinace pravidel 10 a 17 působící po 3 kroky; 1000 kroků

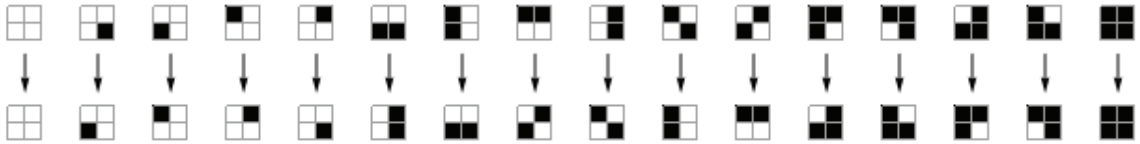


Obrázek 7.53: Kombinace pravidel 10 a 17 působící po 3 kroky; 2000 kroků

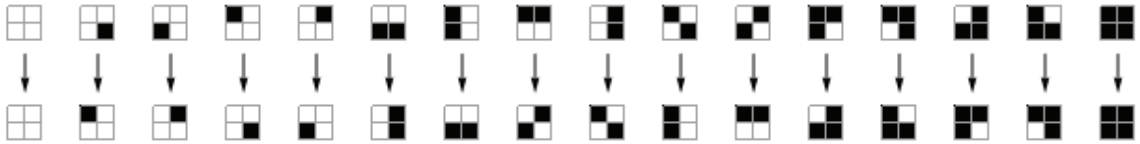


Obrázek 7.54: Kombinace pravidel 10 a 17 působící po 4 kroky; 3000 kroků

roli v případu, kdy jsou částice na počátku koncentrovány v kruhu. Prozkoumali jsme tedy všechna možná pravidla a po zevrubné vizuální analýze jsme zvolili pravidlo číslo 383, jelikož se nám průběh částic pro toto pravidlo jevil jako nejvhodnější. Látka se pro tento případ rozpínala do prostoru se stejnou rychlostí ve všech směrech. Pravidlem pro třetí složku jsme zvolili pravidlo 17. Na Obr. 7.55 a Obr. 7.56 jsou názorně ukázána evoluční transformace pro případ, kdy evoluci daného systém zrovna řídí pravidlo 10-383-17 a pro případ, kdy ho v daném kroku řídí pravidlo 17-383-17.



Obrázek 7.55: Schéma pravidla 10-383-17



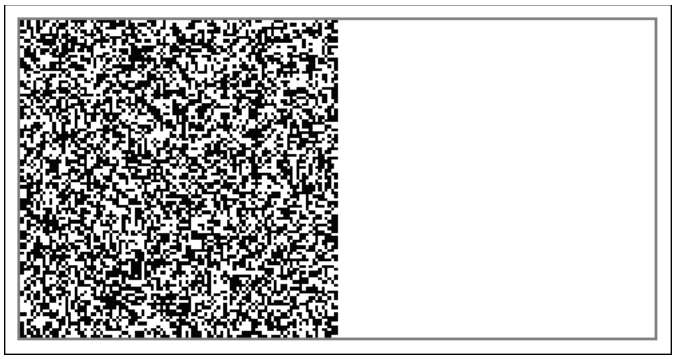
Obrázek 7.56: Schéma pravidla 17-383-17

8. Srovnání nalezeného pravidla s teoretickým modelem

Konečně jsme měli sestavené pravidlo, které by mělo být možné použít pro simulování difúze. Nyní jsme potřebovali zjistit, jak, pokud vůbec, se od sebe liší data naměřená pro naše pravidlo od teoretického modelu difúze, který je popsaný v kapitole 1. V kapitole 1 je podrobně popsán případ, kdy je na počátku měření zaplněna levá polovina trubice (1.6) a také případ, kdy je látce umožněno pohybovat se do obou směrů trubice (1.22).

Chtěli bychom zdůraznit, že náš model je ve své podstatě bezrozměrný. Přesné jednotky délky a času zde zavádí až definice hodnoty konkrétního difúzního koeficientu.

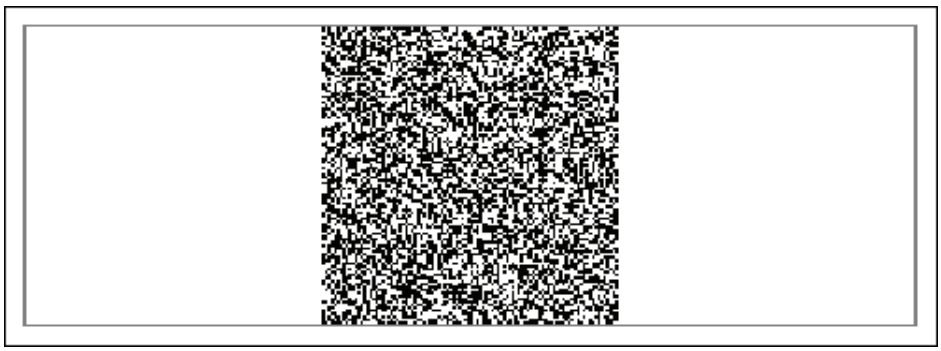
Pro první případ naše experimentální uspořádání vypadalo tak, že jsme měli trubici o rozměrech 200×100 a částice jsme umístili do levé poloviny trubice, takže zabíraly prostor o rozměrech 100×100 (Obr. 8.1). Pro druhý případ jsme použili trubici o roz-



Obrázek 8.1: Počáteční matice pro případ zaplnění trubice z levé strany

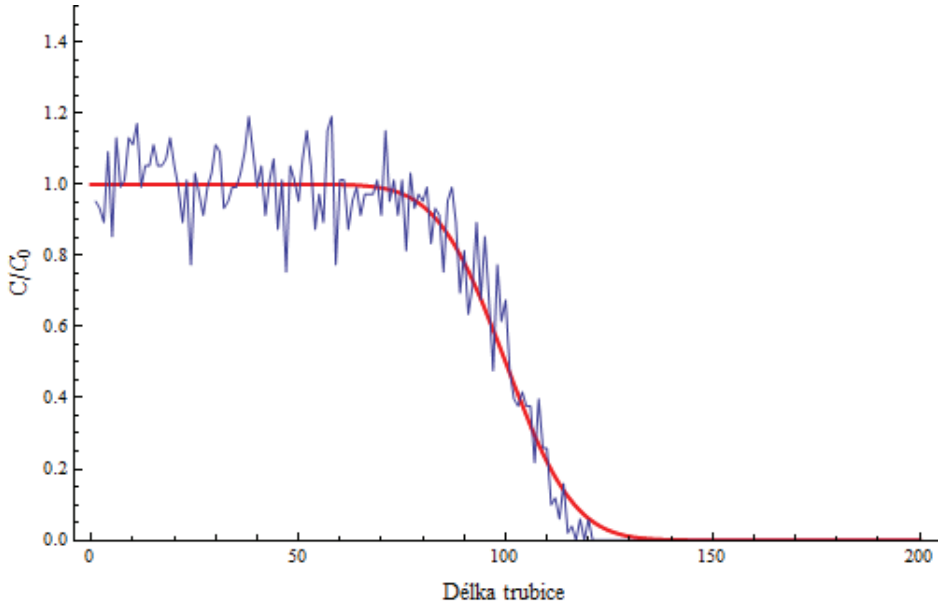
měrech 300×100 a částice opět zaplňovaly prostor o velikosti 100×100, ale tentokrát byly umístěny do středu trubice (Obr. 8.2). Obě trubice byly samozřejmě ohraničeny, čemuž odpovídá šedivě zbarvená čára po obvodu.

Postupovali jsme tak, že jsme vždy na začátku měření zjistili počáteční koncentraci



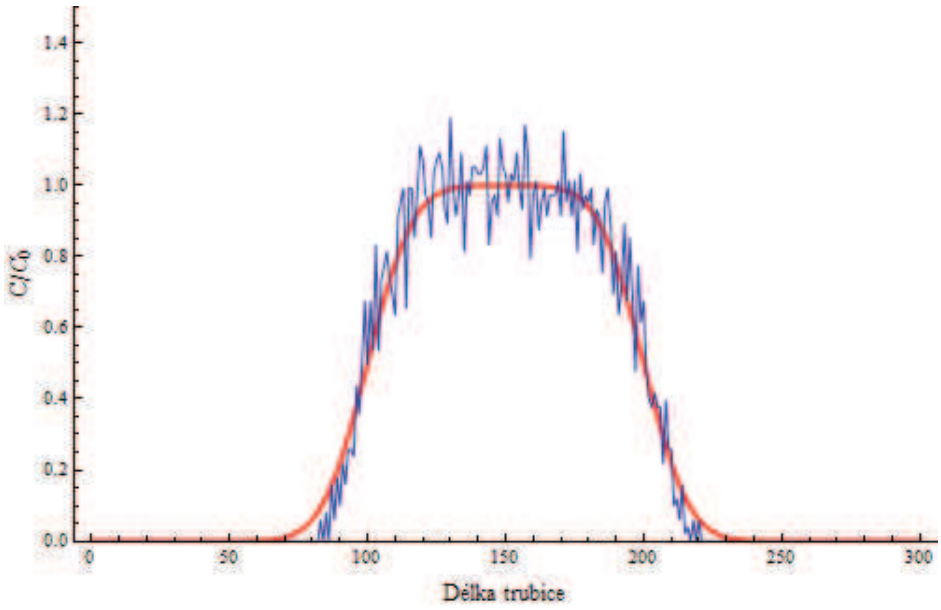
Obrázek 8.2: Počáteční matice pro případ zaplnění trubice v jejím středu

C_0. Tuto hodnotu jsme získali tak, že jsme sečetli všechny částice v prostoru počáteční distribuce a podělili jsme ji počtem buněk, prázdných i plných, v tomto prostoru. Pro oba případy se jednalo o počet 100×100 buněk. Následně jsme pro jednotlivé průběhy vždy měřenou trubici rozdělili na 200, v případě částic ve středu na 300, sloupců, ve kterých jsme sečetli všechny částice a, stejně jako v případě počáteční koncentrace, jsme tuto hodnotu vydělili počtem všech buněk ve sloupci (bylo to vždy 100 buněk). Získali jsme tedy hodnoty koncentrací po celé délce trubice a mohli jsme tak sledovat, jak se postupně, s probíhající evolucí systému, zvyšuje hodnota koncentrace v místech, kde na počátku byla koncentrace nulová. Pro názornost jsme měli vždy na ose y vyneseny hodnoty pro poměr C/C_0. Na Obr. 8.3 a Obr. 8.4 je vidět, že teoretický model a průběh vlastního pravidla se od sebe výrazně neliší.



Obrázek 8.3: Porovnání teoretického průběhu s naměřeným

Aby se náš model dal použít k simulování difúzního pohybu pro konkrétní látku,



Obrázek 8.4: Porovnání teoretického průběhu s naměřeným

je třeba, aby se pro danou látku nakalibrovala doba, po kterou trvá jeden evoluční krok. Každá látka má jinou hodnotu difúzního koeficientu, která závisí na teplotě a jiných veličinách. Proto je zapotřebí zjistit, jak dlouhý časový úsek představuje jeden krok.

Postupovali jsme tak, že jsme si zvolili hodnotu difúzního koeficientu pro plyn vodíku, která při teplotě 273 K a tlaku 0, 1 MPa má hodnotu $1,604\,\mathrm{cm}^2/\mathrm{s}$ [14]. Vzali jsme si data pro případy difúze v trubici, s počáteční distribucí částic na levé straně, pro hodnoty 10, 60, 120, 180, 1200, 3000 a 6000 kroků a ty jsme použili pro stanovení přibližné doby, kterou zabere jeden evoluční krok.

Nejdříve jsme všechny tyto průběhy nafitovali pomocí funkce FindFit, která je v softwaru *Mathematica* obsažena, pro danou hodnotu difúzního koeficientu. Tato funkce používá pro fitování křivek metodu nejmenších čtverců.

Z fitování přes funkci FindFit jsme získali hodnoty času pro vybrané hodnoty kroků a ty jsme vydělili tak, abychom pro každý případ získali dobu pro jeden krok. Pozorovali jsme, že tyto hodnoty se lehce lišily pro každý z vybraných průběhů. Proto jsme z nich udělali aritmetický průměr a získali jsme průměrnou hodnotu času na jeden krok. V Tab. 8.1 jsou zobrazeny hodnoty času pro případ difúze vodíku. Pro vodík nám vyšla průměrná doba, kterou zabere jeden krok, jako 0,868 sekund.

Nyní jsme potřebovali zjistit, jak velmi se od sebe liší křivky získané fitováním pomocí funkce FindFit a křivky, které jsme získali pro jednotlivé průběhy, když jsme jednomu evolučnímu kroku přiřadili hodnotu 0,868 sekund. Postupovali jsme tak,

Krok	Čas získaný fitováním [s]	Přepočet na 1 krok [s]
10	5,60	0,560
60	41,24	0,687
120	85,63	0,714
180	127,40	0,708
1200	750,03	0,625
3000	2501,80	0,834
6000	11711,40	1,950

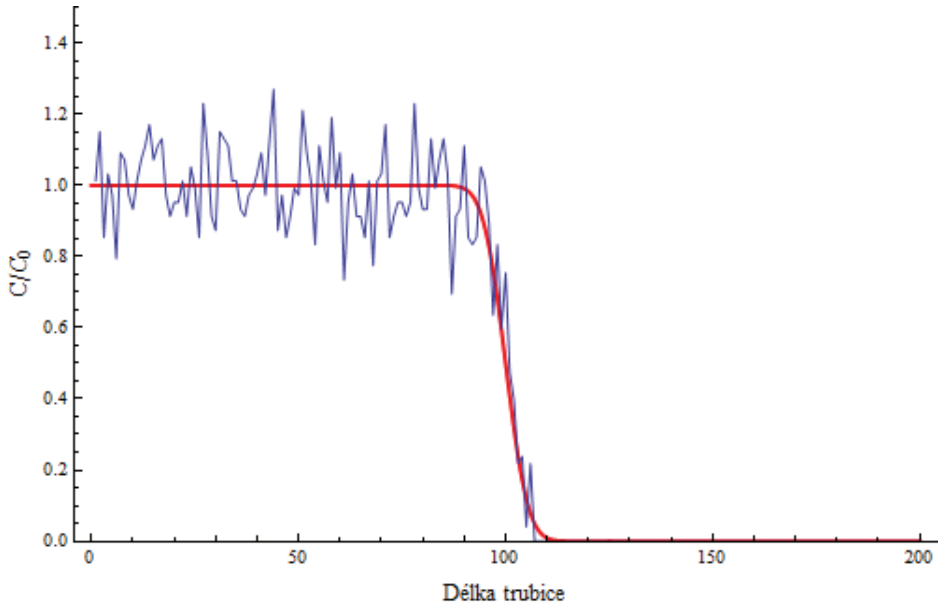
Tabulka 8.1: Doba trvání v sekundách pro jednotlivé kroky

že jsme vždy vypočetli sumu čtverců (RSS) pro naměřená data a dané křivky, kterými jsme je chtěli fitovat, a poté jsme porovnávali takto získané hodnoty pro oba případy, abychom zjistili jak moc se od sebe odlišují. V Tab. 8.2 je vidět, že hodnoty sum čtverců pro případ, kdy jsme průběhy fitovali funkcí FindFit a kdy jsme použili jednotnou dobu pro krok se velmi často liší pouze v řádech desetin až setin.

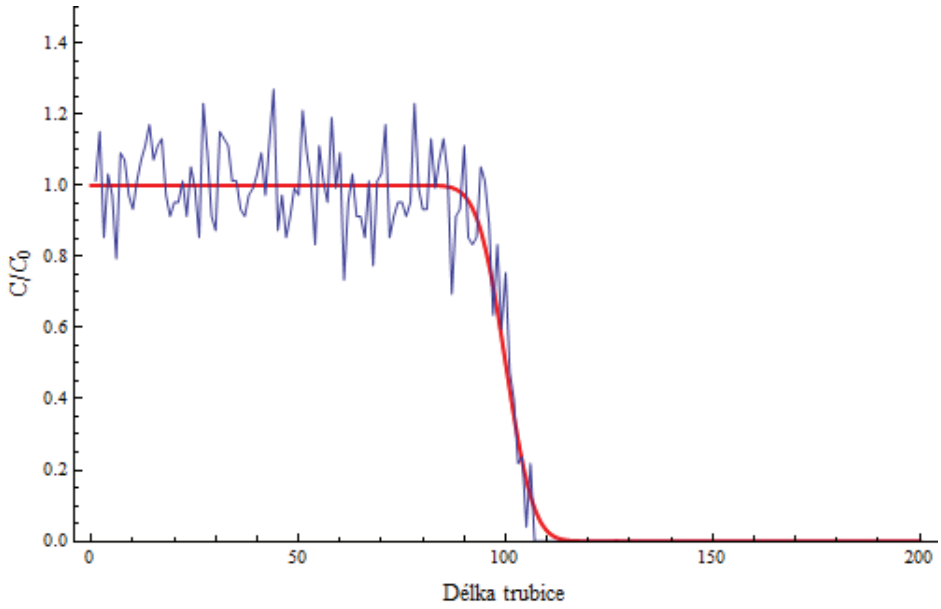
Krok	RSS pro fit	RSS pro jednotný čas
10	1,31982	1,33973
60	0,949184	0,954514
120	1,10247	1,10479
180	0,87186	0,885774
1200	1,23229	1,3044
3000	0,913716	0,912188
6000	0,841596	1,09013

Tabulka 8.2: Hodnoty sum čtverců

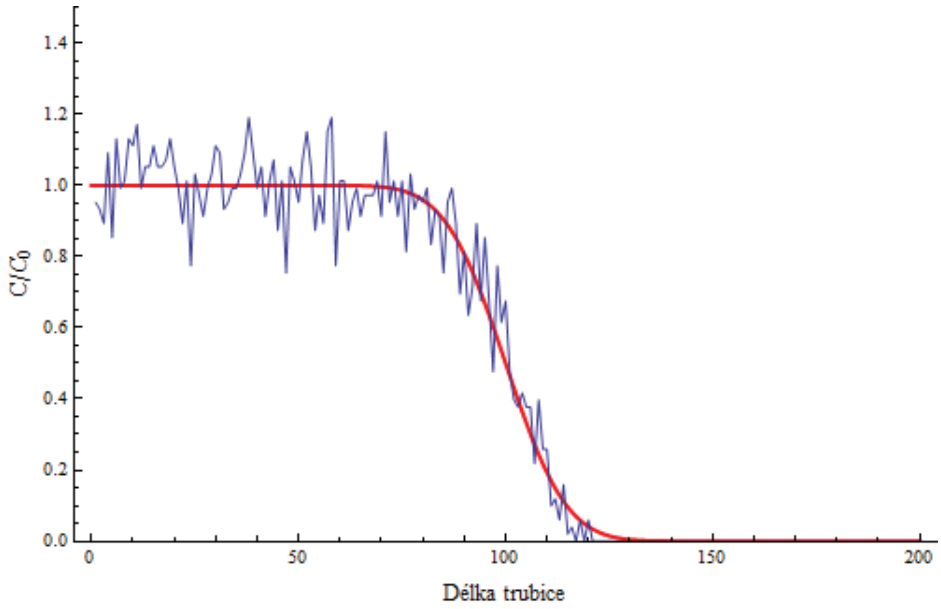
Na Obr. 8.5 až Obr. 8.18 je možné vidět, jak se konkrétně od sebe liší křivky získané funkcí FindFit a křivky při jednotné délce kroku, když je porovnáme s konkrétními naměřenými hodnotami.



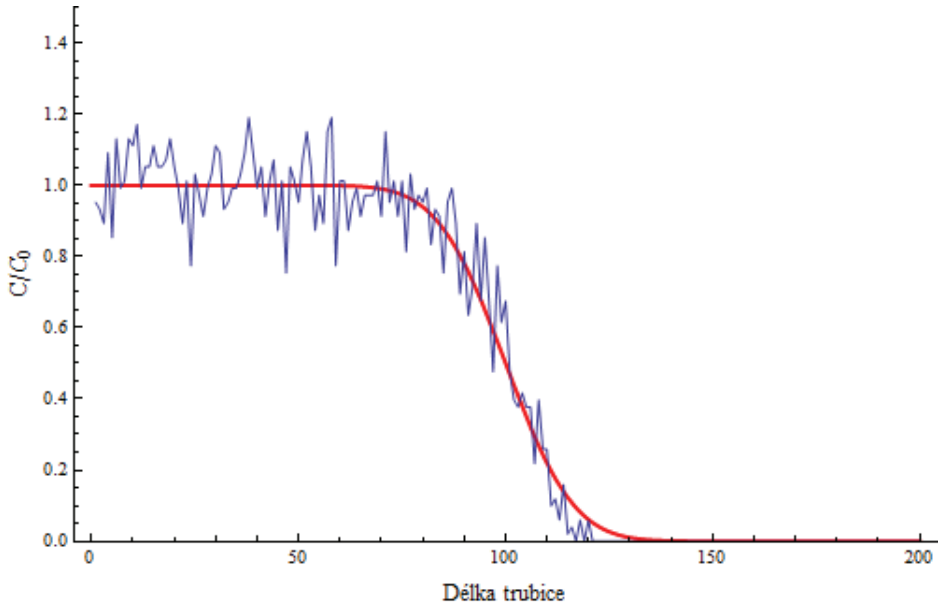
Obrázek 8.5: Krok 10 – fitování



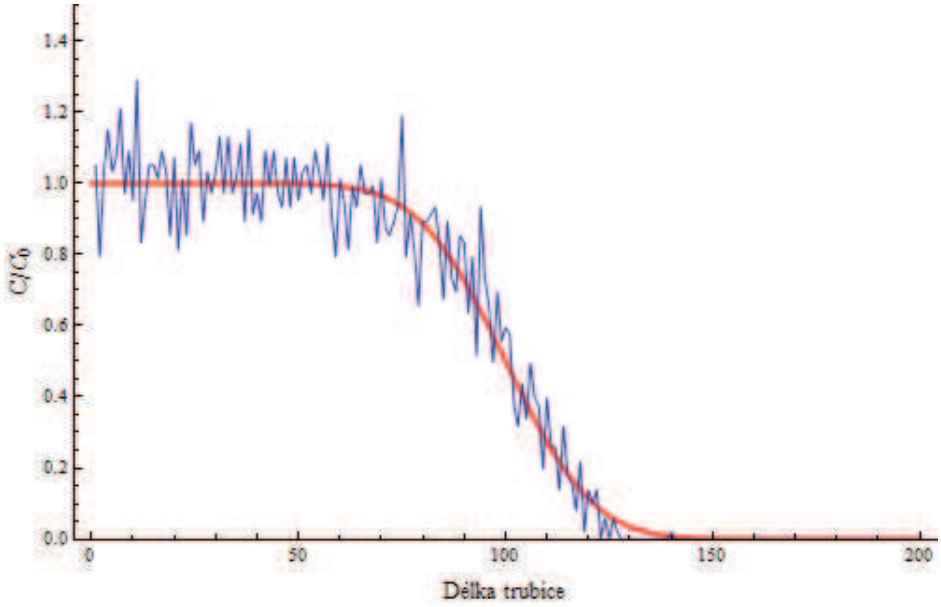
Obrázek 8.6: Krok 10 – jednotná doba kroku



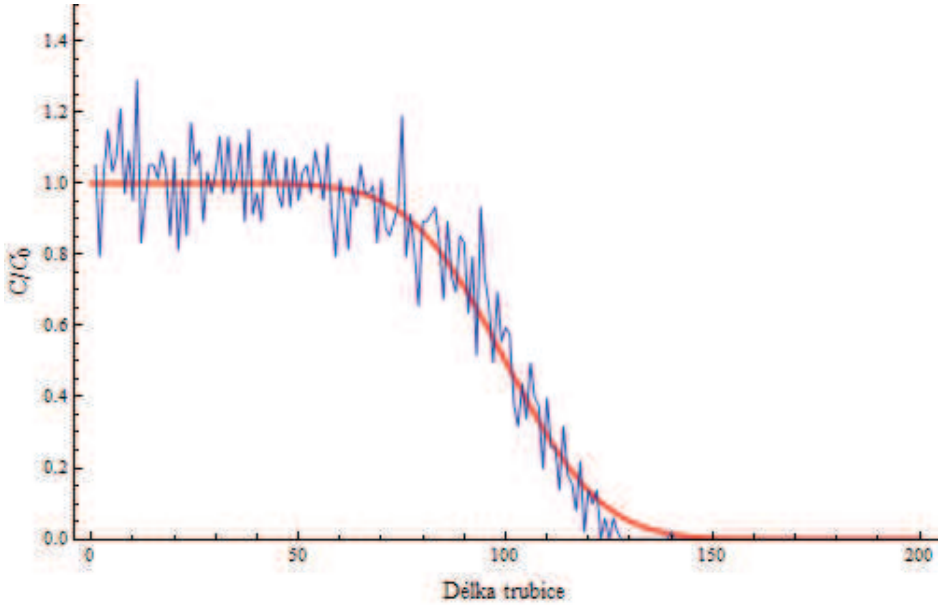
Obrázek 8.7: Krok 60 – fitování



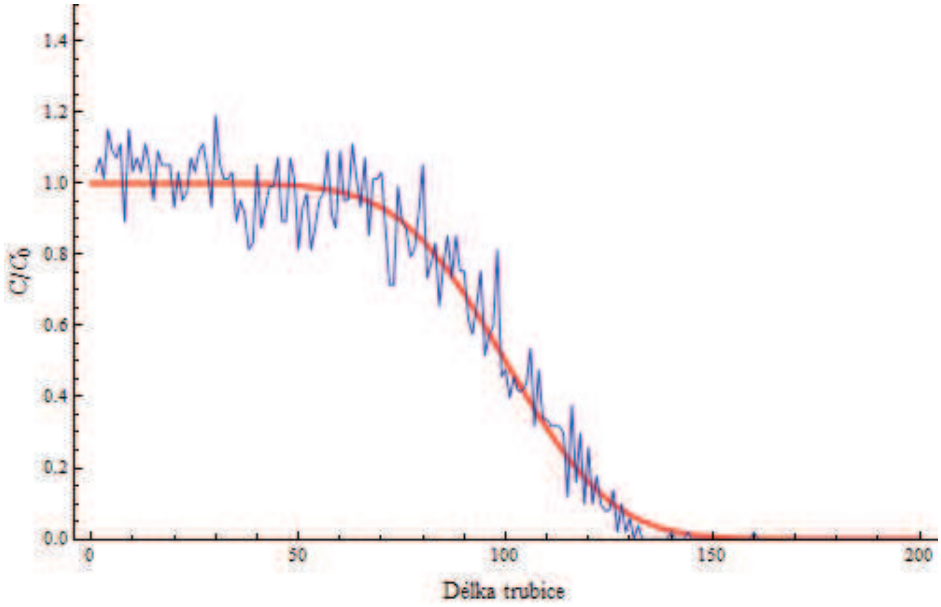
Obrázek 8.8: Krok 60 – jednotná doba kroku



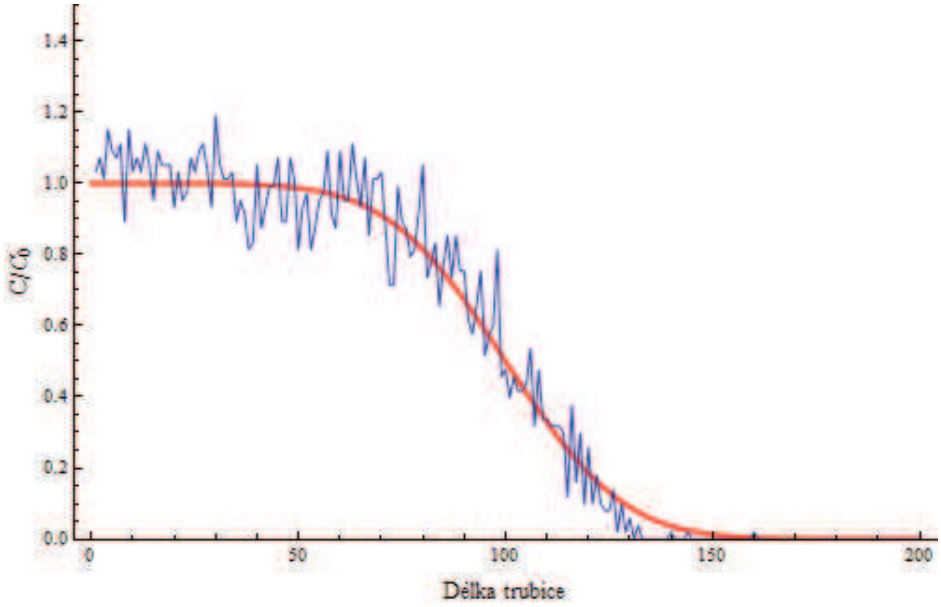
Obrázek 8.9: Krok 120 – fitování



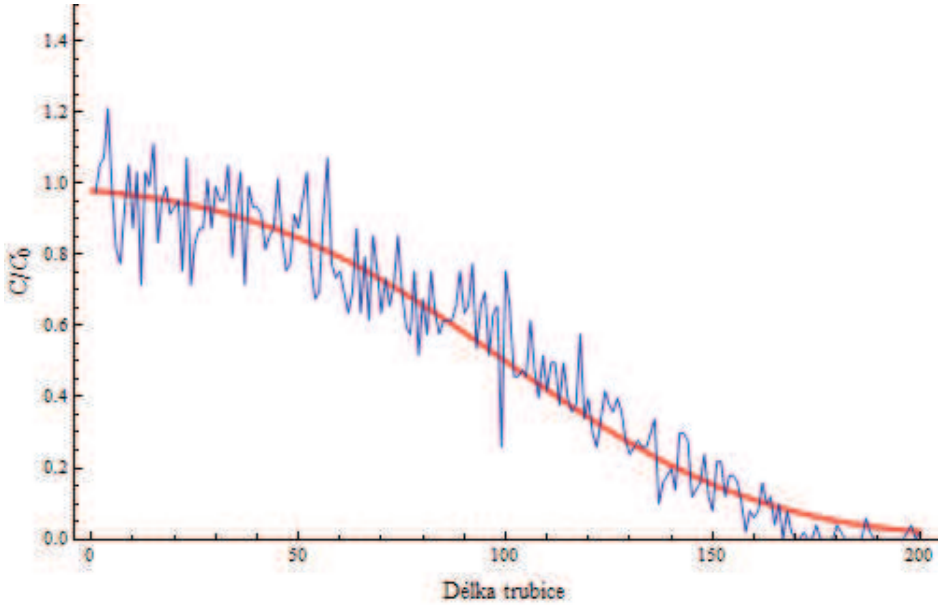
Obrázek 8.10: Krok 120 – jednotná doba kroku



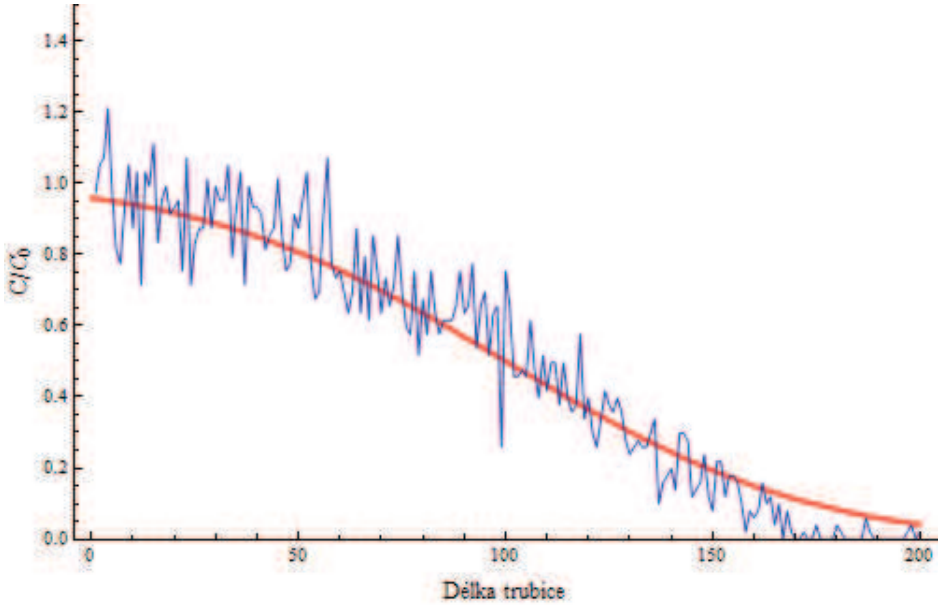
Obrázek 8.11: Krok 180 – fitování



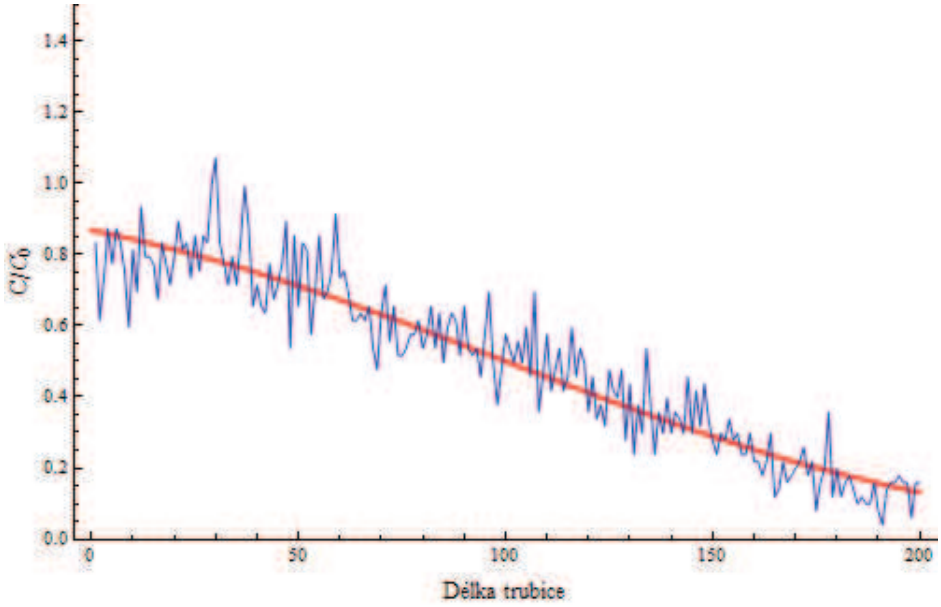
Obrázek 8.12: Krok 180 – jednotná doba kroku



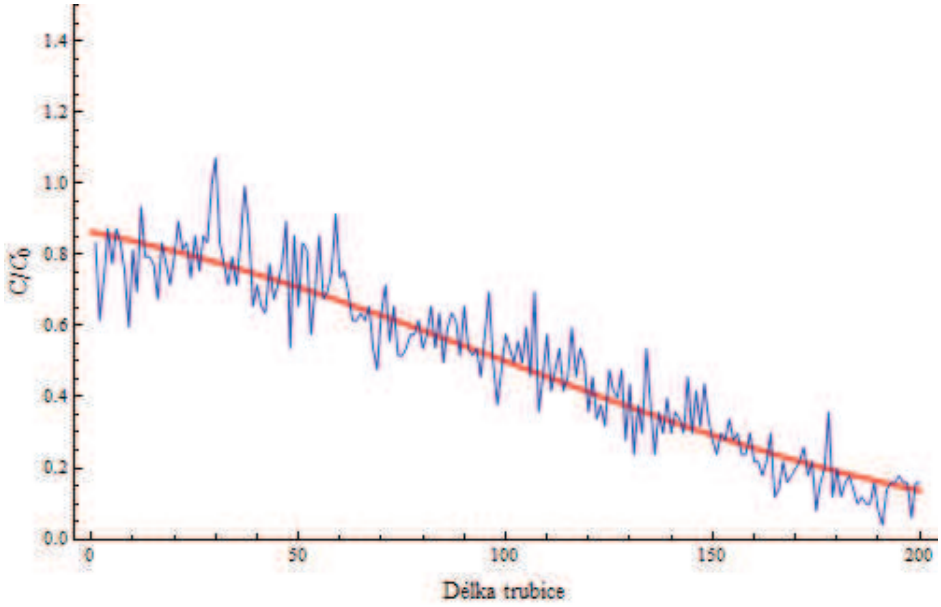
Obrázek 8.13: Krok 1200 – fitování



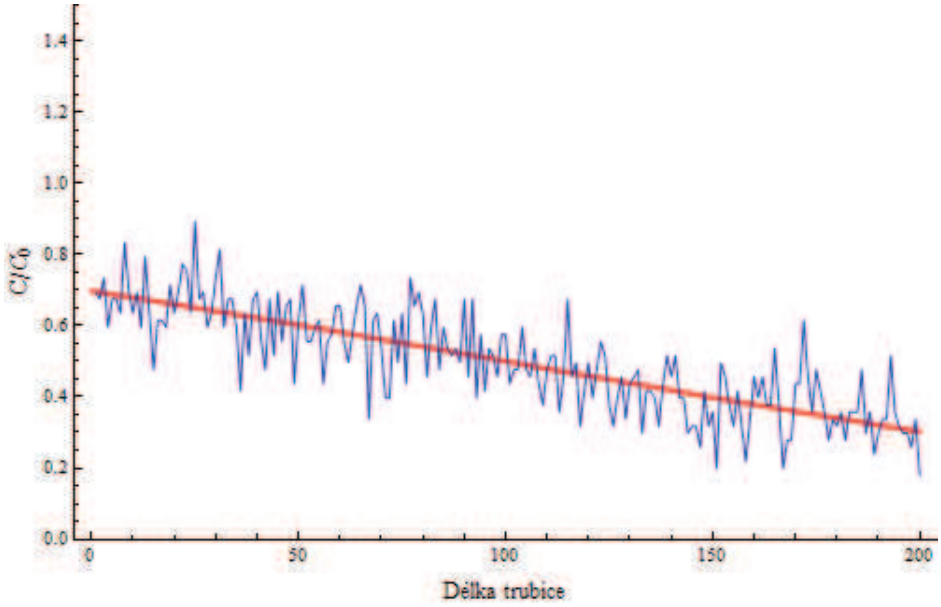
Obrázek 8.14: Krok 1200 – jednotná doba kroku



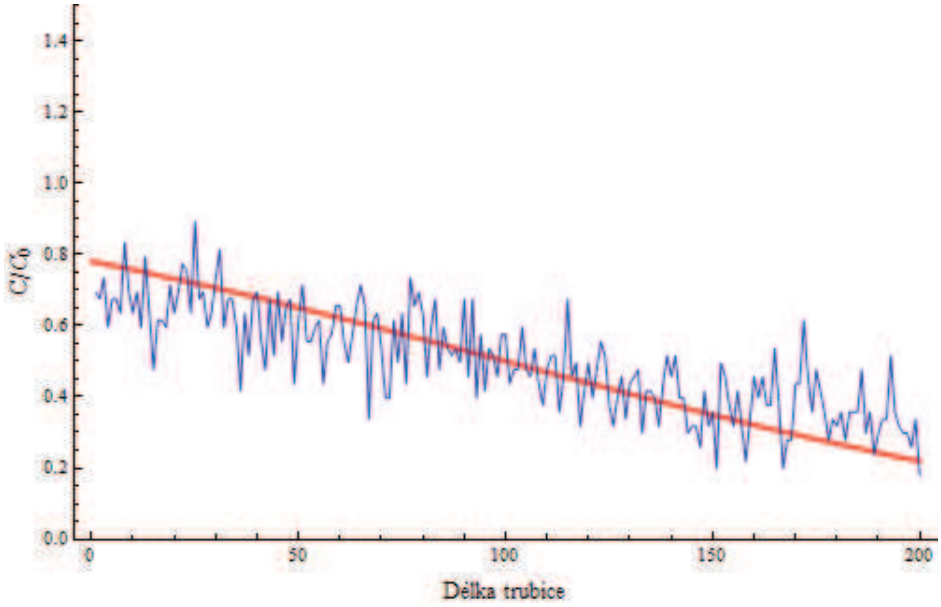
Obrázek 8.15: Krok 3000 – fitování



Obrázek 8.16: Krok 3000 – jednotná doba kroku



Obrázek 8.17: Krok 6000 – fitování



Obrázek 8.18: Krok 6000 – jednotná doba kroku

Závěr

Při prozkoumávání pravidel blokových celulárních automatů jsme objevili pravidlo, přesněji řečeno kombinaci pravidel, která vykazuje, při své evoluci, difúzní pohyb. Toto tvrzení jsme podložili poznatky získanými při porovnávání průběhů této, námi zvolené, kombinace pravidel a teoretického modelu difúze.

Dovolujeme si tvrdit, že naše pravidlo je mnohem vhodnější pro případ, kdy jsou částice umístěny v uzavřeném prostoru, než-li pravidla, která byla při simulování difúze pomocí blokových celulárních automatů používána dosud. Při pouhé rotaci buněk v bloku dochází totiž nevyhnutelně k jevu, kdy se částice koncentrují hlavně podél stěn daného systému, což logicky neodpovídá teorii difúzního pohybu. Naopak naše pravidlo bylo vytvořeno právě s ohledem na to, aby se dalo použít pro uzavřený systém. Věříme, že by se nemuselo omezovat pouze na vodorovný pohyb v trubici, ale měl by obstát i v komplexnějším systému, jako například v soustavě trubic, které by byly vzájemně různě propojené.

Bylo by velmi zajímavé pozorovat průběh našeho pravidla pro systém, jehož rozměry by byly několika set násobně vyšší, než s jakými jsme pracovali v této diplomové práci. Zjistili jsme však, že rozměry tohoto modelu nelze zvyšovat do libovolné velikosti. Hlavní problém zde hraje skutečnost, že při realizaci tohoto modelu v prostředí programu *Mathematica* jsme narazili na určitou hranici, která byla ovlivněná operační pamětí počítače, na kterém jsme výpočty prováděli. Je zde tedy snaha vyzkoušet náš model na mnohem výkonnějším počítači než jsme dosud používali a pokusit se sledovat evoluční průběh systému s co největšími rozměry. Je nutné také počítat s tím, že se zvětšováním rozměrů dochází také k prodlužování doby výpočtu jednotlivých kroků evoluce. Tudíž by se mohlo stát, že výpočet jednoho kroku může zabrat i minuty. V případě, že by výpočet jednoho kroku zabral pouhou minutu, tak by nám, kupříkladu při evoluci po 10 000 kroků, výpočty zabraly něco okolo 6 dnů.

Ačkoliv jsme náš model realizovali v prostředí s menšími rozměry, byli jsme schopni

pozorovat velmi výrazné charakteristické znaky difúzního průběhu. Tudíž i při menších rozměrech je náš model schopen velmi přesně simulovat difúzní pohyb.

Literatura

[1] GHOSH, K., DILL, K.A., INAMDAR, M.M., SEITARIDOU, E., PHILLIPS, R. Teaching the principles of statistical dynamics. In *American Journal of Physics*, 2006, roč. 74, s. 123-133.

[2] SLUTSKY, M. Diffusion in a half-space: From Lord Kelvin to path integrals. In *American Journal of Physics*, 2005, roč. 73, s. 308-314.

[3] OLSZEWSKI, E.A. From baking a cake to solving the diffusion equation. In *American Journal of Physics*, 2006, roč. 74, s. 502-509.

[4] COLLINS, R., CARSON, S.R., MATTHEW, J.A.D. Diffusion equation for one-dimensional unbiased hopping. In *American Journal of Physics*, 1997, roč. 65, s. 230-237.

[5] Diffusion. In *Wikipedia* [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006. Poslední aktualizace 19. 4. 2013 [cit. 2013-03-12]. Dostupné na: `http://en.wikipedia.org/wiki/Diffusion`

[6] CRANK, J. *The Mathematics of Diffusion*. 2. vyd. Oxford: Clarendon press, 1975.

[7] RAY, E., BUNTON, P., POJMAN J.A. Determination of the diffuson coefficient between corn syrup and distilled water using a digital camera. In *American Journal of Physics*, 2007, roč. 75, s. 903-906.

[8] CHOPARD, B., DROZ, M. *Cellular Automata Modeling of Physical Systems*. Cambridge: University Press. 1998.

[9] WOLFRAM, S. *A New Kind of Science*. Champaign: Wolfram Media, 2002.

[10] BANDMAN, O.L. Comparative Study of Cellular-Automata Diffusion Models. In *Parallel Computing Technologies*, 1999, roč. 1662, s. 395-409.

[11] CHOPARD, B., DROZ, M. Cellular Automata Model for the Diffusion Equation. In *Journal of Statistical Physics*, 1991, roč. 64, s. 859-892.

[12] WELLIN, R.P., GAYLORD, J.R., KAMIN, N.S. *An Introduction to programming with Mathematica*. 3. vyd. Cambridge: University Press, 2005.

[13] DICK, S., RIDDLE, A., STEIN, D. 1997. *Mathematica in the laboratory*. Cambridge: Cambridge University Press, 1997.

[14] Mostinsky, I.L. Diffusion coefficient. In *Thermopedia* [online]. 2. 2. 2011, Poslední aktualizace 10. 2. 2011 [cit. 2013-04-01]. Dostupné na: `http://www.thermopedia.com/content/696/`

Seznam obrázků

Seznam tabulek