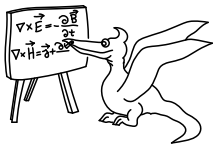


Seriál: Numerické metody a počítačové simulace

31. ročník FYKOSu a s ním i seriál o numerických metodách a simulacích vstupují do druhé poloviny. Je proto na čase opustit metodu Monte Carlo a věnovat se zcela novému tématu – buněčným automatům. Seznámíme se s jejich principy a ukážeme si několik jednoduchých příkladů jejich použití. V numerické části se budeme věnovat klasickému, ale velmi užitečnému tématu: hledání kořenů funkcí.

Hledání kořenů

Hledáním kořenů funkce $f(x)$ máme na mysli hledání všech nebo některých řešení rovnice $f(x) = 0$. Předpokládáme přitom, že f je reálná funkce reálného argumentu, tedy že x i hodnoty $f(x)$ jsou reálná čísla (a ne komplexní čísla, vektory, ...). Dále předpokládáme, že funkce je spojitá (její graf lze nakreslit jedním tahem), pro některé metody pak musíme předpokládat i spojitost prvních derivací (graf funkce nemá ostré „zuby“, takové funkce se proto označují jako hladké). Je poměrně zřejmé, že jde o rozumné předpoklady, které naprostá většina funkcí, se kterými jste se na střední škole setkali, splňuje.

Ve škole jste se většinou setkali pouze s rovnicemi, které dokážete analyticky vyřešit. Ve skutečnosti je ale takových rovnic poměrně málo.[1] Jako příklad analyticky neřešitelných rovnic poslouží třeba $x = \cos x$ nebo řešení úlohy Jáchymovská z letošního FoLu. Hledání kořenů má ale i jiné aplikace, například může posloužit k numerickému nalezení inverze funkce. Pokud totiž pro zvolené y chceme najít x takové, že $y = f(x)$, pak nejde o nic jiného, než právě hledání kořenu funkce $g(x) = f(x) - y$. Věříme, že jsme vás již dostatečně namotivovali, pojďme si proto nějaké základní metody představit.

Metoda půlení intervalu

Na začátku si potřebujeme zvolit dvě hodnoty a_0 a b_0 tak, že v intervalu (a_0, b_0) leží právě jeden kořen (ten, který chceme najít). Pak vezmeme střed tohoto intervalu, tedy hodnotu $c_0 = (a_0 + b_0)/2$ a podíváme se, jestli kořen leží v intervalu (a_0, c_0), nebo (c_0, b_0) (nebo jestli není $f(c_0)$ zrovna nula, i když díky omezené přesnosti reálných čísel tento případ typicky nenastává). Interval, ve kterém leží kořen, pak použijeme jako interval (a_1, b_1) a iterujeme pro stále větší n, dokud nebude interval (a_n, b_n) dostatečně malý. Pak je kořen přibližně roven hodnotě $(a_n + b_n)/2$ s maximální chybou $(b_n - a_n)/2$.

Algoritmus máme téměř hotov, chybí zjistit, jak rozhodnout, ve kterém z intervalů kořen leží. Využijeme zde Bolzanovy věty, která říká: „Nechť f je spojitá na omezeném a uzavřeném intervalu $\langle a, b\rangle$ a $f(a)f(b) < 0$. Pak existuje (alespoň jedno) $c \in (a, b)$ takové, že $f(c) = 0$.“ Rozmyslete si, že věta říká poměrně intuitivní věc: „Pokud $f(a)$ a $f(b)$ mají rozdílné znaménko a spojím je jedním tahem (grafem funkce), pak tento tah někde protíná osu x.“ Věta nám tedy napovídá, že správným testem je kontrolovat jestli $f(a_i)f(c_i) < 0$, nebo $f(c_i)f(b_i) < 0$ (pokud $f(c_i) \neq 0$, bude alespoň jedna nerovnost platit). Všimněte si ale, že věta nám neříká, že kořen bude v intervalu pouze jeden. To musíme zajistit na začátku, proto jsme si původní

[1] Ve smyslu, že pokud náhodně vymyslíme rovnici, tak ji vyřešit nedokážeme.

interval zvolili tak, aby v něm ležel právě jeden kořen. Pokud toto nesplníme, pak může metoda samozřejmě selhat – pokud pro počáteční interval platí $f(a_0)f(b_0) < 0$, nějaký kořen nalezneme vždy, ale jiné se můžou ztratit.

Metoda půlení intervalu (též bisekce) je velmi jednoduchá metoda, její základní myšlenka se tedy ujala i v problémech, které s hledáním kořenů nesouvisí. Jako příklad poslouží třeba hledání hodnoty v setříděném poli, kdy se nejprve koukneme doprostřed, pokud je hodnota menší, koukneme se na prvek v jedné čtvrtině délky pole, pokud je větší, na prvek ve tří čtvrtinách, ... Je pozoruhodné, že se tato metoda osvědčila i jako „dřevorubecký způsob“ hledání některých typů chyb v programu. Pokud nám např. nějaká funkce dává chybný výsledek, stačí si postupně vypisovat hodnoty z poloviny metody, čtvrtiny metody, ... a ověřovat, jestli je tam hodnota výpočtu ještě v pořádku. Tímto způsobem pak poměrně rychle dokážeme najít konkrétní příkaz, kde chyba nastává.

Z popisu algoritmu je patrné, že bisekce (stejně jako všechny ostatní metody, které zde představíme) je iterativní metoda. Nikdy nedostaneme zcela přesný výsledek, ale čím více iterací provedeme, tím více se mu budeme blížit. Důležitou vlastností iterativních metod je rychlost konvergence, tedy to, jak rychle se budeme správné hodnotě blížit s rostoucím počtem iterací. Nyní se pokusme tento pojem formalizovat. Mějme posloupnost $\{x_n\}_{n=0}^{\infty}$ konvergující k hodnotě s. Pokud existují konečné konstanty $k \neq 0$ a $p > 0$ takové, že

$$\lim_{n\to\infty} \frac{|x_{n+1} - s|}{|x_n - s|^p} = k\,,$$

pak řekneme, že posloupnost konverguje s *řádem konvergence* p. Speciálně pokud $p = 1$, pak posloupnost konverguje lineárně, pokud $p = 2$, pak konverguje kvadraticky, ...

Pokud bude konvergující posloupností posloupnost chyb metody (v závislosti na počtu iterací) $\{e_n\}_{n=0}^{\infty}$, kde $e_n = |x_n - s|$, pak dle definice bude pro danou posloupnost pro dostatečně veliké n platit $e_{n+1} \approx k e_n^p$. Tento rekurentní vztah můžeme rozepsat do podoby

$$e_n \approx k^{1+p+\cdots+p^{(n-n_0-1)}} e_{n_0}^{p^{(n-n_0)}}\,,$$

kde jsme si zvolili pevné dostatečně velké n_0 a uvažujeme $n \geq n_0$. Zlogaritmováním pak dostaneme

$$\ln e_n \approx p^{n-n_0} \ln e_{n_0} + \sum_{j=0}^{n-n_0-1} p^j \ln k\,.$$

Pro případ $p > 1, k < 1, e_{n_0} < 1$ tedy platí, že závislost *logaritmu* chyby na n se bude chovat přibližně jako $-Cp^n$, kde $C > 0$ je konstanta. Pro $p = 1$ se logaritmus chyby chová jako $-Cn$, chyba tedy klesá exponenciálně.

S rychlostí konvergence jsme se již setkali v úloze o hledání čísla π v první sérii. Ve vzorovém řešení bylo zmíněno, že metoda konverguje kvadraticky s počtem hran, a názorně jsme si předvedli, co to znamená. Upozorněme zde ale, že šlo o jinou definici rychlosti konvergence, která je používána spíše pro diskretizační metody (metody, kde kouskujeme nějakou oblast či čas na malé dílky). V obou definicích ale platí, že čím vyšší řád, tím lépe metoda konverguje, jen to pokaždé znamená něco trochu jiného.

Vypočtěme nyní řád konvergence pro metodu bisekce. Je zřejmé, že v $n$tém kroku hledáme kořen v intervalu šířky $|b_n - a_n| = 2^{-n}|b_0 - a_0|$. Protože v tomto intervalu kořen jistě leží, můžeme touto hodnotou shora odhadnout chybu metody e_n v $n$tém kroku. Posloupnost $\{e_n\}$ konverguje k nule, dle výše uvedené definice tedy hledáme takové $p > 0$, aby výraz

$$\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^p} = \lim_{n\to\infty} \frac{2^{-n-1}|b_0 - a_0|}{(2^{-n}|b_0 - a_0|)^p} = \frac{1}{2}|b_0 - a_0|^{1-p} \lim_{n\to\infty} \left(2^{p-1}\right)^n$$

byl roven nenulovému reálnému číslu. Je vidět, že pro $p > 1$ je hodnota výrazu rovna $+\infty$, pro $p \in (0, 1)$ je pak rovna nule. Řešením je $p = 1$, kdy je hodnota výrazu rovna číslu $1/2$. Dokázali jsme tedy, že metoda konverguje lineárně.

Pokud si to tedy shrneme, pro metodu bisekce potřebujeme dopředu znát interval, kde se kořen vyskytuje a jde o lineárně konvergující metodu. Lze si také rozmyslet, že metoda konverguje vždy (což, jak uvidíme, není samozřejmost). Jde tedy o jednoduchý a robustní algoritmus, ne ale příliš rychlý. Představme si tedy rychleji konvergující metody.

Metoda sečen

Myšlenka této metody je opět poměrně jednoduchá. Na začátku potřebujeme dva body x_0 a x_1. Kořen nemusí tentokrát nutně ležet mezi nimi, musí ale ležet „dostatečně blízko“. Nyní proložíme body $[x_0, f(x_0)]$ a $[x_1, f(x_1)]$ přímku – sečnu funkce $f(x)$. Ta je v jistém smyslu rozumnou aproximací funkce $f(x)$ blízko bodů x_0 a x_1, navíc je to přímka, a pro ni dokážeme najít kořen lehce. Označme tento kořen sečny x_2. Protože je ale sečna pouze aproximací, x_2 (většinou) není kořenem funkce $f(x)$. Proto proložíme další sečnu body $[x_1, f(x_1)]$ a $[x_2, f(x_2)]$, najdeme kořen této sečny x_3 a dále iterujeme až do dosažení konvergence. Tu můžeme detekovat například tak, že se dvě následující hodnoty x_i v podstatě neliší.

Bez důkazu uveďme, že řád konvergence metody je roven zlatému řezu, tedy přibližně 1,62, což je lepší než u lineárně konvergující bisekce. Problém ale je, že metoda nekonverguje vždy. Jako příklad si uveďme funkci, která je na nějakém intervalu konstantní. Pokud se v nějaké iteraci stane, že chceme proložit sečnu dvěma body ležícími v tomto intervalu, nebude mít tato sečna žádný kořen. To je samozřejmě extrémní případ, stačí, pokud je funkce někde málo strmá (v porovnání se strmostí v okolí kořenu funkce), pak sice sečna protne osu x, ale zpravidla v bodě ležícím daleko od skutečného kořenu funkce.

Metoda regula falsi

Pokusme se modifikovat metodu sečen tak, aby byla vždy konvergentní. Na začátku si opět zvolíme x_0 a x_1, tentokrát ale tak, aby kořen ležel mezi nimi (aby $f(x_0)f(x_1) < 0$). Těmito dvěma body opět proložíme sečnu a najdeme její kořen x_2. Nyní ale nastane podstatná změna. Další sečnu proložíme buď body x_1 a x_2, nebo x_0 a x_2 podle toho, mezi kterou dvojicí bodů leží skutečný kořen funkce $f(x)$, což, podobně jako u bisekce, ověříme podle toho, jestli $f(x_0)f(x_2) < 0$, nebo $f(x_1)f(x_2) < 0$. Poté iterujeme až do požadované přesnosti.

Vidíme, že podstatnou modifikací je to, že si hlídáme, aby skutečný kořen ležel vždy mezi průsečíky sečny s osou x. Tento požadavek nám zajistí, že se odhad kořene x_i bude přibližovat skutečnému kořenu a metoda tedy bude vždy konvergentní. Zaplatíme za to ale tím, že metoda konverguje pouze lineárně, nicméně o něco rychleji než bisekce (ne o tolik, aby se to projevilo na řádu konvergence, ale i tak je to vítané zlepšení).

Newtonova metoda tečen

Myšlenka Newtonovy metody je taková, že zvolíme nějaký bod x_0 „blízko“ hledaného kořenu funkce $f(x)$. Funkci poté aproximujeme tečnou v bodě x_0, což technicky provedeme rozepsáním Taylorova polynomu v okolí bodu x_0 do 1. řádu

$$f(x) \approx y = f(x_0) + f'(x_0)(x - x_0)\,.$$

Nyní, podobně jako v předchozích dvou metodách, nalezněme kořen této aproximace, tedy položme $y = 0$. Pak platí

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}\,,$$

protože je ale toto x opět pouhou aproximací kořenu, označme jej jako x_1 a iterujme dále. Kompletní rekurentní předpis Newtonovy metody tedy je

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}\,.$$

Specifikem této metody je, že je potřeba znát nejen funkční hodnoty samotné funkce $f(x)$, ale i její derivace. Mohli byste namítnout, že pokud bychom derivaci neznali, mohli bychom použít její numerickou aproximaci. Nicméně lze ukázat, že v případě použití dopředné diference přechází Newtonova metoda na metodu sečen (zkuste si to, jde jen o dosazení do vzorce). To dává smysl, neboť numerická derivace je pouze aproximací, hledáme tedy vlastně sklon sečny procházející body x a $x + h$, zatímco při použití derivace je h infinitesimálně malé, sečna tedy přejde v tečnu.

S metodou sečen má Newtonova metoda společnou i tu nehezkou vlastnost, že nekonverguje vždy. Pokud ale konverguje, tak konverguje (za splnění určitých předpokladů) kvadraticky, jde tedy o nejefektivnější z metod, které jsme si zde představili.

Banachova věta o kontrakci

Jako poslední si představme metodu, kterou nelze použít obecně, nicméně za určitých okolností se může vyplatit ji znát. Navíc jde o poměrně zajímavou matematickou větu, se kterou jste se již možná setkali, ani o tom nevíte.

Banachova věta je poměrně obecné tvrzení, které se zavádí na abstraktním matematickém objektu nazývaném metrický prostor. My se zde ale spokojíme s méně obecnou verzí této věty, kdy si ji zavedeme pouze na intervalech reálných čísel, což, pokud přirozeně použijeme absolutní hodnotu rozdílu dvou čísel jako tzv. metriku, je speciální případ metrického prostoru.

Nejprve si ale potřebujeme definovat několik pojmů. Funkce $f : \langle a, b\rangle \to \mathbb{R}$ se nazve *lipschitzovská*, pokud existuje $L \geq 0$ takové, že $\forall x, y \in \langle a, b\rangle : |f(x) - f(y)| \leq L|x - y|$. Pokud navíc $L < 1$, pak funkci f na intervalu $\langle a, b\rangle$ nazveme *kontrakcí*. Toto označení má jasný význam, neboť funkční hodnoty kontrakce jsou si blíže, než původní argumenty, kontrakce tedy interval $\langle a, b\rangle$ zobrazuje na interval $\langle c, d\rangle$, který je menší, než původní interval, „zkontrahuje jej“.

Nyní již samotná Banachova věta. Nechť $f : \langle a, b\rangle \to \langle a, b\rangle$ je kontrakce. Pak existuje jediné $x_\mathrm{P} \in \langle a, b\rangle$ takové, že $f(x_\mathrm{P}) = x_\mathrm{P}$ (x_P je tzv. *pevný bod zobrazení*). Navíc pokud $x_0 \in \langle a, b\rangle$ a $x_{k+1} = f(x_k)$, pak pro všechna $k = 1, 2, \ldots$ platí

$$|x_k - x_\mathrm{P}| \leq \frac{L^k}{1 - L}|x_0 - x_1|\,.$$

Takto matematicky napsaná věta nejspíš vypadá hrozivě, pojďme si tedy rozebrat, co říká. První část tvrdí existenci právě jednoho bodu, který se kontrakcí zobrazí sám na sebe. Nejlépe si to představíme na příkladu. Hezkým příkladem kontrakce je turistická mapa, která zobrazuje reálné body ve světě na papír. Banachova věta pak tvrdí, že pokud mapu položíme na zem někde

v oblasti, kterou zobrazuje,[2] pak existuje na mapě právě jeden bod, který se zobrazuje sám na sebe. Druhá část věty pak říká, že pokud vezmeme libovolný bod z intervalu $\langle a, b\rangle$ a budeme na něj opakovaně aplikovat kontrakci f, dokonvergujeme právě do pevného bodu zobrazení. Věta navíc říká, jak rychle tam dokonvergujeme (jde vidět, že jde o lineární konvergenci).

Zde se dostáváme k aplikaci. Zkusili jste si někdy na kalkulačce zadat nějaké číslo a pak opakovaně mačkat tlačítko `cos`? Všimli jste si přitom, že ať už zadáte jakékoliv číslo, vždy nakonec dospějete k číslu přibližně 0,739? Pak jste použili právě Banachovu větu o kontrakci a nalezli tak kořen rovnice $x = \cos x$, neboť funkce cos je kontrakce. Počáteční hodnotu jste mohli volit kdekoliv, protože je to kontrakce na libovolném intervalu obsahujícím interval $\langle -1, 1\rangle$.

Nevýhodou Banachovy věty je, že funguje pouze na kontrakce, musíte tedy mít štěstí na řešený problém. V praxi tedy nejspíš použijete některou z ostatních metod. Jde ale rozhodně o pozoruhodný a v určitých oblastech matematiky i velmi důležitý koncept.

Celulární automaty

Mnohé matematické a fyzikální systémy jsou popsány rovnicemi, které nedokážeme analyticky řešit a zapojujeme proto do práce numeriku a simulace. Existují však i situace, kdy je systém příliš komplikovaný na to, abychom dokázali všechny potřebné rovnice vůbec sepsat. O to překvapivější je, když takový systém po čase vykazuje zřejmé symetrie, opakující se geometrické vzorce, vyvíjí se od chaosu k jednoduchému uspořádání. Příkladem může být růst sněhové vločky či krystalu obecně. Ačkoli je agregace částic velice komplikovaný a tedy těžko předvídatelný proces, výsledkem je geometricky přesný krystal. Jinými slovy, jednoduché globální uspořádaní je důsledkem složitého působení lokálních interakcí.

Aniž bychom příliš zabíhali do termodynamiky, rozmyslíme si, že jestliže entropie („neuspořádanost“) izolovaného systému samovolně neklesá, musí systémy vykazující samouspořádání být otevřené.[3] Naše neschopnost snadno popsat takový systém soustavou rovnic s jednoznačným řešením plyne z toho, že by tato soustava musela obsahovat časový vývoj v principu nekonečného prostředí. Chceme tedy najít model sestávající z rozumně malého počtu pravidel, který dokážeme prostudovat detailněji než velkou soustavu diferenciálních rovnic (popisující stejný problém) a zároveň bude vykazovat netriviální dynamické chování vedoucí, pro určité hodnoty parametrů, k samoorganizaci. Jak jsme již předeslali, cílem tohoto textu bude ukázat, že právě takový model nám poskytnou buněčné (celulární) automaty.

Se studiem celulárních automatů (CA) začali matematičtí fyzikové Stanisław Ulam a John von Neumann během pobytu v Los Alamos za druhé světové války. Ulamovou motivací bylo zkoumání růstu krystalů pomocí mřížkových modelů, zatímco von Neumanna k CA přivedla myšlenka autoreplikace strojů. Společně pak svoje znalosti použili ke studiu pohybu tekutin pomocí metody, která spočívala v prostorové diskretizaci tekutiny a zkoumání interakce mezi sousedními buňkami. V sedmdesátých letech se do širšího povědomí dostal 2D buněčný automat zvaný Hra života, který objevil John Conway.[4] Tento automat je zajímavý množstvím statických i dynamických objektů, které se v něm i přes jeho jednoduchá pravidla objevují, a ještě se k němu později vrátíme. V osmdesátých letech se do práce na buněčných automatech pustil Stephen

[2] To je důležitý požadavek, vyjádřený matematicky $f : \langle a, b\rangle \to \langle a, b\rangle$. Pokud mapu Prahy položíme na zem v New Yorku, nejsou předpoklady Banachovy věty splněny.

[3] Otevřený termodynamický systém je takový, který si s okolím může vyměňovat hmotu i energii. O okolí se nezajímáme (neznáme jeho extenzivní popisné veličiny jako je objem, hmota, entropie), pouze měříme toky skrze hranice studovaného systému. Izolovaný systém si nevyměňuje ani hmotu, ani energii.

[4] Anglický matematik, narozen 1937.

Wolfram,[5] přičemž rigorózně klasifikoval všechny jednorozměrné automaty a pomohl rozšířit CA jako metodu do dalších oborů, nejen přírodovědných.

Vlastnosti a klasifikace CA

Celulární automat je jednoznačně definován pomocí tří objektů. První z nich je (pro účely simulací konečná, matematicky je možné studovat i nekonečné) *mřížka* Λ s pozičním indexem i, která nám říká, jakým způsobem jsme diskretizovali prostor. Může být čtvercová, hexagonální i jakákoli jiná. Dále musíme zadat *množinu stavů* S, kterých může nabývat každá jednotlivá buňka. Stav konkrétní buňky označíme s_i. Nakonec musíme znát lokální *pravidlo*, podle kterého se systém v diskrétním čase vyvíjí. Takové pravidlo lze obecně napsat ve tvaru

$$s_i(t+1) = f(\{s\}_{O(i)}(t))\,. \tag{1}$$

Zde máme na levé straně stav buňky na pozici i v čase $t+\Delta t$, přičemž jsme pro jednoduchost volili $\Delta t = 1$. Napravo máme funkci f, která nabývá hodnot z množiny S. Jejím argumentem jsou stavy buněk z okolí $O(i)$ v čase t. Okolí můžeme definovat libovolně, obvykle se jedná o nejbližší sousedy, samotná buňka i může, ale nemusí být do okolí zahrnuta.

Ještě než si ukážeme příklad jednoduchého pravidla f, povězme si něco stručně o klasifikaci buněčných automatů. Podle charakteru pravidla můžeme CA dělit na *deterministické* a *stochastické*. V prvním případě je na pravé straně rovnice (1) obyčejná funkce, zatímco stochastické pravidlo má na pravé straně funkci vracející náhodnou proměnnou (která může nabývat hodnot pouze z množiny S). Dále můžeme CA rozdělit podle toho, jak aplikují své pravidlo. Pokud při provádění jednoho časového kroku vkládáme do funkce napravo vždy hodnoty buněk, kterých nabývali před provedením tohoto kroku, jedná se o *synchronní* automat. V jakémkoli jiném případě nazýváme automat *asynchronním*. Striktně řečeno jsme tedy výše definovali pouze deterministický a synchronní automat, to proto, abychom se vyhnuli komplikovanému zápisu při větším zobecnění. Dalším možným kritériem pro klasifikaci CA je jejich komplexita, která sahá od typů, kdy vždy po pár krocích dosáhneme statického stavu, až po automaty s velmi dlouhým dynamickým vývojem. Užitečným kritériem klasifikace je také stabilita vůči poruše.

A nyní k příkladu. Uvažujme 1D automat (tvar mříže zde není potřeba udávat) s binární množinou stavů $S = \{0, 1\}$ s pravidlem

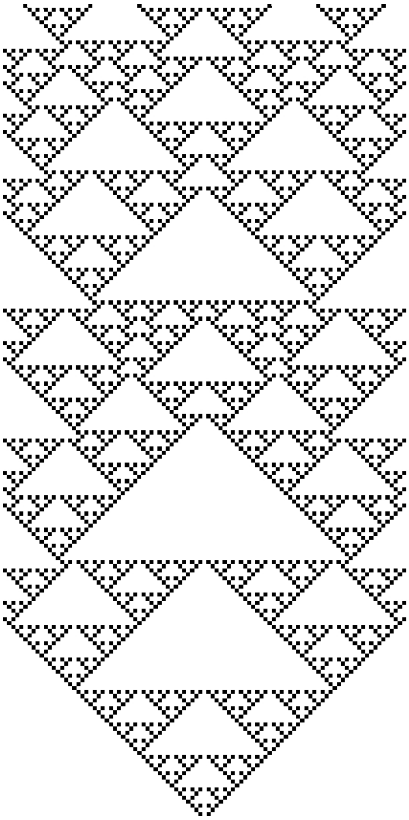
$$s_i(t+1) = s_{i-1}(t) \veebar s_{i+1}(t)\,, \tag{2}$$

kde symbol $\veebar$ označuje logický operátor výlučné nebo (exclusive or, xor).[6] Při časovém vývoji se tedy u každé buňky díváme, jakých hodnot nabývají nejbližší sousedi. Pokud právě jeden z nich má hodnotu 1 (uvažujeme synchronní automat, tedy v čase t), pak bude mít buňka na pozici i v čase $t+1$ hodnotu 1, jinak 0. Jelikož je mřížka Λ omezená, je potřeba stanovit, jak se chovají buňky na okraji. Obvykle předpokládáme periodické podmínky (sousedem poslední buňky je první buňka), ale nic nám nebrání volit i jiné. Obrázek 1 nám ukazuje prvních 200 kroků vývoje. Všimněte si, že se v obrázku opakují podobné geometrické útvary, ale nejedná se o periodické chování.

Jednorozměrné automaty je pro jejich jednoduchost poměrně snadné klasifikovat. Budeme uvažovat $O(i) = \{i-1, i, i+1\}$, tedy do okolí zahrnujeme i buňku na aktuální pozici. Také se

[5] Anglický počítačový fyzik, narozen 1959. Ano, je to *ten* Wolfram. Jeho zájem o celulární automaty se projevil i v online algebraickém systému Wolfram|Alpha, který ve svých starších verzích míval na čekací obrazovce během výpočtu animaci vývoje některých CA.

[6] Od klasického *nebo* (or) se liší tím, že pro $A = 1$, $B = 1$ platí $A \vee B = 1$, ale $A \veebar B = 0$.



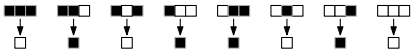
Obr. 1: 1D buněčný automat na mřížce délky 100 s periodickými podmínkami a s pravidlem (2). Na počátku má jeden bod hodnotu 1 a ostatní 0 (počáteční stav není vyobrazen). Na svislé ose je čas (rostoucí směrem nahoru), na vodorovné index i.

omezíme na binární množinu S. Potom může celé okolí nabývat osmi možných stavů: 111, 110, 101, 100, 011, 010, 001, 000. Pro každý z těchto stavů existují dvě možnosti, jak se bude buňka i vyvíjet: buďto bude mít v příštím kroce hodnotu 1, nebo 0. Celkem tedy existuje 256 různých pravidel[7] pro automaty uvedeného typu (občas se nazývají *elementární buněčné automaty*). Pravidlo (2) potom nazveme „pravidlo 90“, jak je názorně ukázáno na obrázku 2.

Podívejme se, jak se to má s počtem automatů ve více dimenzích. Pokud budeme za okolí $O(i)$ opět považovat pouze nejbližší sousedy a samotnou buňku i, bude okolí v d dimenzích obsahovat $N = 2d + 1$ buněk. Velikost množiny stavů je obecně $|S| = q$, počet možných konfigurací na okolí je tedy $C = q^N$. Z každé konfigurace může vzejít q různých stavů buňky i, a proto celkový počet pravidel bude v tomto obecném případě

$$N_{\mathrm{CA}} = q^C = q^{q^{2d+1}} . \tag{3}$$

[7] S využitím symetrií tento počet můžeme redukovat na 32.



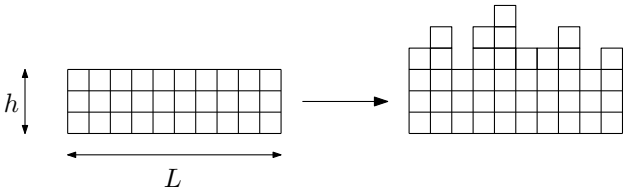
$$01011010 = 2^6 + 2^4 + 2^3 + 2^1 = 90$$

Obr. 2: Názorná ukázka klasifikace pravidla daného rovnicí (2).

Tak například pro dvourozměrný binární buněčný automat existuje celkem $2^{2^{4+1}} = 4\,294\,967\,296$ pravidel. Při takovém počtu nelze pravidla klasifikovat jedno po druhém, smysl má pouze dívat se na statistické chování. Prozatím proto zůstaneme v 1D světě a podíváme se na jednu zajímavou fyzikální aplikaci.

Využití CA pro modelování růstu povrchů

Představme si, že máme kovovou destičku, jejíž povrch je dokonale rovný. Tedy, na jejím povrchu jsou atomy uspořádány jeden vedle druhého, jak je ukázáno v levé části obrázku 3. Na tuto destičku naprášíme[8] několik vrstev stejného materiálu. Nový materiál se však neusazuje zcela rovnoměrně, po naprašování vypadá nanesená vrstva například jako na pravé části obrázku 3. Destička není již tak hladká jako na počátku – říkáme, že se zvýšila *hrubost povrchu*.



Obr. 3: Ukázka hrubnutí 1D povrchu.

Nechť kovová mřížka destičky neobsahuje žádné dislokace. Potom se na ni můžeme dívat jako na jednotlivé sloupečky. Výšku sloupečků částic značme h. Hrubost povrchu definujeme jako odmocninu druhého centrálního momentu výšky[9]

$$W(t,L) = \sqrt{\mathrm{E}(h^2) - (\mathrm{E}(h))^2}\,. \tag{4}$$

Rozepsáno podrobněji,

$$W(t,L) = \sqrt{\frac{1}{L^d}\sum_i h_i^2 - \left(\frac{1}{L^d}\sum_i h_i\right)^2}\,, \tag{5}$$

kde t je čas, L je charakteristický lineární rozměr povrchu, d je jeho dimenze a h_i výška sloupečku s indexem i. Budeme-li se dívat na řez destičkou, bude dimenze povrchu $d = 1$ a L bude délka

[8] Naprašování je jednou z metod epitaxe neboli depozice tenkých vrstev. Zdrojový materiál po malém množství převádíme do plynného stavu a částice plynu pomocí elektromagnetického pole deponujeme na destičku.

[9] Směrodatná odchylka výšky, chcete-li.

řezu (počet sloupečků). Pokud navíc budeme pokládat rozměr molekuly za jednotkový, udává h počet molekul v každém sloupečku a L je počet molekul v řadě.

S časem a přibývajícími částicemi na povrchu hrubost roste. Ale jak? To záleží na způsobu depozice částic, respektive na metodě, kterou ho budeme modelovat. My budeme k simulaci používat 1D asynchronní buněčný automat, přičemž počet buněk bude L a výšky sloupců částic budou představovat jejich stavy. Formálně je tedy množina stavů $S = \mathbb{N}$ neomezená, prakticky ale samozřejmě nevytvoříme destičku nekonečné tloušťky.

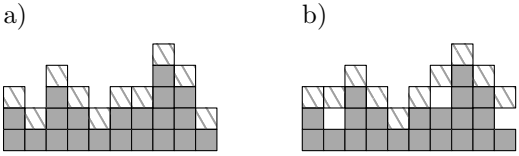
Pravidla evoluce automatu pro simulaci růstu hrubosti mohou být různá. Nejjednodušším pravidlem je *náhodná depozice*, tj. v každém kroku (máme asynchronní automat, krok znamená, že přibude jedna částice, ne celá vrstva) vybereme jednu buňku a zvýšíme její hodnotu o jedna. Náhodný výběr děláme z rovnoměrného rozdělení na L, není-li fyzikální důvod provádět výběr jinak.

Dalším používaným modelem růstu je *balistická depozice*. Opět vybíráme rovnoměrně náhodně buňky, které budou růst, ale v tomto případě se pomyslná padající částice nemusí zastavit až na vybraném sloupečku, ale může se „přilepit“ na sloupeček sousední. Při vybrání buňky i tedy můžeme formulovat evoluční vztah (s vhodnou úpravou na okraji)

$$h_i(t+1) = \max\left(h_{i-1}(t),\, h_i(t)+1,\, h_{i+1}(t)\right) . \tag{6}$$

Ve struktuře destičky tedy vznikají dutiny, automat si je ale nepoznamenává.

Názorná ukázka toho, jak se chovají uvedené dva modely, je na obrázku 4.



Obr. 4: a) Model náhodné depozice, b) model balistické depozice. **Nejedná se o časový vývoj**. Schémata pouze zobrazují, kam dopadne nová (šrafovaná) částice, bude-li vybrán určitý sloupec/buňka. Kdyby se jednalo o evoluci, mohly by se nové buňky přilepovat i vzájemně na sebe.

Další příklady CA

V povídání o 1D automatech jsme se nevěnovali příliš jednotlivým pravidlům, byť jsou některé z nich nepochybně zajímavé. Například[10] u již zmíněného pravidla 90 existují pro každou konfiguraci čtyři možní předchůdci (zde uvažujeme mřížku bez okrajů, nekonečnou; tvar okrajových podmínek má taky vliv na počet předchůdců), zatímco u některých jiných CA existují konfigurace, ke kterým se nelze evolucí dostat. Pravidlo 110 zase plynule přechází mezi chaotickými a uspořádanými oblastmi v časo-prostorovém diagramu a navíc je turingovsky úplné.[11]

[10] Mějte na paměti, že ke každému příkladu existuje několik symetrických automatů, striktně řečeno tedy nejsou unikátní.

[11] Být turingovsky úplný ve stručnosti znamená být schopný řešit všechny úlohy, které dokáže řešit jakýkoli jiný stroj. Neznamená to ale řešit efektivně, viz 8-bitové procesory sestavené v Minecraftu...

Mnohem zajímavější chování nalezneme mezi 2D automaty. Již výše zmíněná Hra života (Game of Life, GoL) je synchronní, deterministický, binární CA s následujícím pravidlem:

$$\begin{aligned} s_i(t+1) &= 1 \quad \text{pro} \quad 1 < |O(i)| < 4 \\ &= 0 \quad \text{jinak}\,. \end{aligned} \tag{7}$$

Název Hra života plyne z představy, že buňka s více než třemi sousedy umírá v důsledku „přelidnění“ a buňka s jedním nebo žádným sousedem umírá z osamění. Toto jednoduché pravidlo vede ke vzniku mnoha statických i dynamických vzorů, navíc je GoL dokáže nekonečně replikovat. Na rozdíl od pravidla 90 existují v GoL konfigurace, ke kterým není možno dospět evolucí – takové počáteční stavy se obvykle velice rychle redukují na základní objekty. Také nelze na základě počátečního stavu jednoznačně předpovědět, které objekty budou během evoluce vznikat. To mimo jiné plyne z toho, že GoL je podobně jako pravidlo 110 turingovsky úplná (kdyby šlo bez proběhnutí evoluce předpovědět výsledek, byla by GoL „silnější“ než univerzální Turingův stroj, což není možné). Nejlépe se s GoL seznámíte, když si sami vyzkoušíte nechat vyvíjet některé počáteční stavy. Ke hraní doporučujeme online simulátor `https://bitstorm.org/gameoflife/`.

Hra života je zajímavá, ale není vidět její přímý užitek. Za užitečné můžeme považovat například modelování tvorby kolon v dálničním provozu. Úplně nejjednodušším modelem je pravidlo 184, kde buňka s hodnotou 1 představuje auto a buňka s hodnotou 0 prázdné místo. Pokud je napravo od auta prázdné místo, tak se pohne, jinak stojí (rozmyslete si, jak formálně toto pravidlo může vypadat, případně si to dopočtěte pomocí schématu). Pokročilejší model je Nagelův-Schreckenbergův, stochastický 1D celulární automat s proměnlivou rychlostí „auta“. Nebudeme ho zde podrobně rozebírat, pouze uvedeme zajímavý výsledek: kolona se s průběhem času posouvá proti směru rychlosti aut.

Pomocí buněčných automatů dokážeme simulovat (a bylo to nakonec i naší motivací) také problémy, které nejde dobře fyzikálně uchopit. Pěkným příkladem je CA model šíření lesních požárů. Jedná se o 2D stochastický, synchronní automat, v němž každá buňka nabývá několika stavů z množiny hodnot mladý strom, starý strom, hořící strom, spáleniště (dělení může být i podrobnější), přičemž požár se s nejvyšší pravděpodobností rozšíří na starý strom, s menší na mladý strom a nemůže se rozšířit na spáleniště. Ze spáleniště se v každém časovém kroku může s určitou pravděpodobností stát mladý strom. V případě tohoto automatu závisí výsledek velice silně na poměrech zadaných pravděpodobností. K vyhrání si doporučujeme online simulátor `http://www.eddaardvark.co.uk/svg/forest/forest.html`.

Celulární automaty se také hodí ke studiu kritických jevů, jakými jsou například fázové přechody. O tom ale až příště.

Fyzikální korespondenční seminář je organizován studenty MFF UK. Je zastřešen Oddělením pro vnější vztahy a propagaci MFF UK a podporován Ústavem teoretické fyziky MFF UK, jeho zaměstnanci a Jednotou českých matematiků a fyziků.