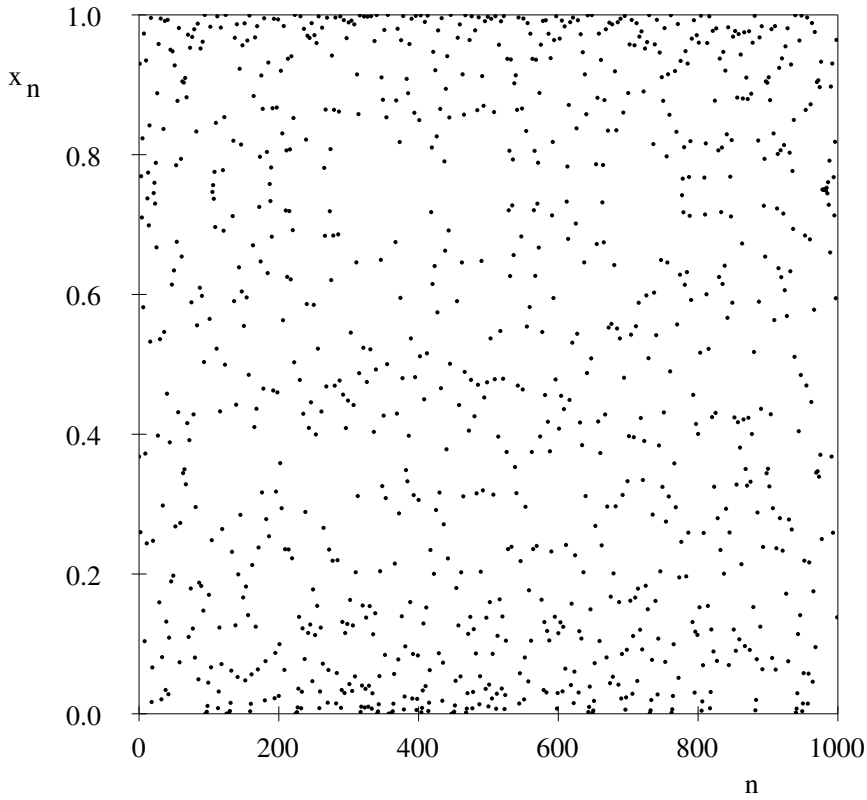
# ÚSKALÍ ČÍSLICOVÝCH POČÍTAČŮ PŘI SIMULACI NELINEÁRNÍCH DYNAMICKÝCH SYSTÉMŮ

**Pavel Pokorný**

Současné číslicové počítače používají pro uchování a zpracování přibližných desetinných čísel dvojkovou soustavu. Proč právě dvojkovou? Bylo by možné použít jinou, např. trojkovou nebo desítkovou soustavu?

Ano, bylo, ale dvojková soustava je výhodná pro zjednodušení konstrukce hardwaru a také pro omezení výskytu chyb, protože pravděpodobnost, že bude daná hodnota nějaké fyzikální veličiny, která se používá pro reprezentaci informace v paměti počítače, např. elektrického napětí, která je nutně zatížená šumem, chybně vyhodnocena, je tím větší, čím bližší jsou hodnoty, tedy tím větší, čím více hodnot se používá. Dvě hodnoty jsou minimum, pravděpodobnost chyby je pak nejmenší. Např. při použití desítkové soustavy by byla vzdálenost sousedních hodnot přibližně desetkrát menší, tedy pravděpodobnost chyby by byla větší. Použití dvojkové soustavy má ale i závažné nevýhody při simulaci dynamických systémů.




Obr. 1: Prvních tisíc členů posloupnosti generované chaotickým dynamickým systémem (1).

Uvažujme, dnes již klasický, dynamický systém

$$x_{n+1} = f(x_n), \qquad f(x) = 4x(1-x), \tag{1}$$

s počáteční podmínkou např.

$$x_1 = 1/\mathrm{e} \doteq 0{,}367879. \tag{2}$$

Obrázek 1 ukazuje prvních tisíc členů posloupnosti $x_n$, které nabývají hodnot mezi nulou a jedničkou zdánlivě nepravidelně. Lze spočítat Lyapunovův exponent

$$\lambda = \ln 2,$$

takže se jedná o deterministických chaos. Že se opravdu jedná o deterministický proces se lze snadno přesvědčit i bez znalosti rovnic, z kterých byla postoupnost vypočtena. Stačí si vynést do grafu v rovině body o souřadnicích $(x_n, x_{n+1})$ a uvidíme, že leží na křivce, v tomto případě parabole, což je graf funkce, jejíž iterací posloupnost vznikla. Potud je vše v souladu s očekáváním na základě zkušeností z teorie dynamických systémů.

Systém (1) lze homeomorfismem

$$h(x) = \sin^2(x\frac{\pi}{2}), \qquad x \in \langle 0, 1\rangle$$

převést na systém

$$y_{n+1} = g(y_n), \qquad g(y) = 1 - |2y - 1|. \tag{3}$$

Snadno se lze přesvědčit, že pro $x \in \langle 0, 1\rangle$ platí

$$f \circ h = h \circ g, \tag{4}$$

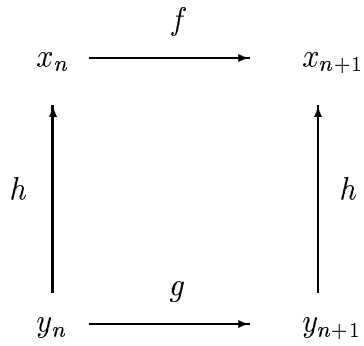
tedy

$$f(h(x)) = h(g(x)),$$

tedy vlastně

$$4\sin^2(x\frac{\pi}{2})(1 - \sin^2(x\frac{\pi}{2})) = \sin^2((1 - |2x - 1|)\frac{\pi}{2}),$$

což lze vyjádřit grafem na Obr. 2. Pak říkáme, že $f$ a $g$ jsou konjugované homeomorfismem $h$.




Obr. 2: Komutativní diagram ilustrující vztah (4). Říkáme, že dynamické systémy (1) a (3) jsou topologicky ekvivalentní.

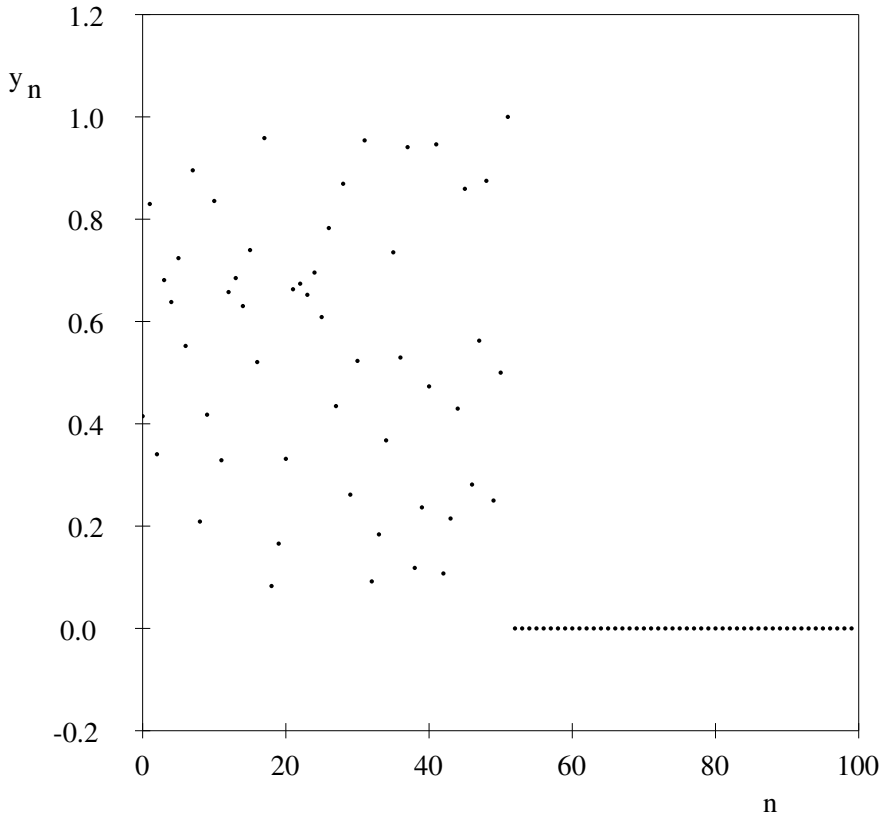
A tedy platí i

$$f^n \circ h = h \circ g^n \qquad n \in N, \tag{5}$$

čili

$$f^n = h \circ g^n \circ h^{-1} \qquad n \in N. \tag{6}$$

Takže libovolnou iteraci $f$ lze vyjádřit pomocí příslušné iterace $g$ a obráceně. Speciálně, má-li systém (1) periodickou orbitu s periodou $p$, pak má i systém (3) periodickou orbitu s touž periodou.




Obr. 3: Prvních sto členů posloupnosti generované chaotickým dynamickým systémem (3). Po nejvýše 53 iteracích dostaneme pro libovolnou počáteční podmínku při simulaci na číslicovém počítači samé nuly. To je nepříznivý důsledek toho, že čísla jsou v počítači uložena ve dvojkové soustavě a toho, že derivace funkce $g$ v (3) je (v bodech, kde derivace existuje) přesně rovna 2.

Jak dopadne numerická simulace systému (3) na číslicovém počítači? Počáteční podmínce (2) odpovídá počáteční podmínka

$$y_1 = \frac{2}{\pi} \arcsin \sqrt{\frac{1}{\mathrm{e}}} \doteq 0{,}414879. \tag{7}$$

Obr. 3 ukazuje výsledek simulace systému (3). Prvních cca 50 iterací vypadá podobně jako na Obr. 1. Ale od jistého $n$ jsou hodnoty $y_n$ konstantní, totiž nulové. To je překvapivé, protože podle (4) jsou systémy (1) a (3) ekvivalentní. Příčina je v tom, že hodnoty $y_n$ jsou uloženy v počítači jako přibližná desetinná čísla v dvojkové soustavě, viz. Tab. 1.

Způsob uložení čísel v paměti počítače a práce s nimi záleží na použitém softwaru. V běžných programovacích jazycích, jako je C, Fortran, Pascal atd. se nejčastěji používá tzv. dvojitá přesnost, kdy je přibližné desetinné číslo uloženo v semilogaritmickém tvaru pomocí 8 bytů, tedy 64 bitů. Těchto 64 bitů je rozděleno do třech skupin:

- 1 bit znaménko: 0 znamená plus, 1 znamená mínus,
- 11 bitů exponent,
- 52 bitů mantisa (dvojková místa za desetinnou čárkou).

Hodnota čísla v tomto semi-logaritmickém tvaru pak je

$$x = \text{znaménko } (1+\text{mantisa}) \times 2^{\text{exponent - 1023}}.$$

Vyjímku tvoří číslo nula, které je uloženo jako samé nuly. Např. na řádku 49 v Tab. 1 vidíme: znaménkový bit je 0, tedy plus, exponent je $01111111110_2 = 1022$ a mantisa je $11_2$ (dalších 50 nul nevypisujeme), takže desetinná hodnota je

$$(1 + 0{,}11_2) \times 2^{1022-1023} = 0{,}875.$$

Rozsah 52 dvojkových míst mantisy znamená relativní přesnost

$$2^{52} \doteq 10^{16}$$

tedy cca 16 desetinných míst a maximální exponent 2047 znamená, že největší číslo v tomto formátu, s kterým lze pracovat, je přibližně

$$2 \times 2^{2047-1023} \doteq 4 \times 10^{308},$$

což pro většinu aplikací postačuje. Pro větší nároky můžeme použít čtyřnásobnou přesnost nebo speciální software (Mathematica, Maple), tím však výrazně klesá rychlost výpočtu.

Podívejme se nyní na tento náš výsledek z určitého nadhledu. Při vyčíslování pravé strany dynamického systému (3) se číslice ve dvojkovém zápise posouvají doleva (a případně transformují) a zprava se doplňují nulami. To je zvláštní případ. Jak to bude vypadat při simulaci obecného dynamického systému? K posunu informace z míst daleko za desetinnou čárkou směrem doleva, tedy do míst s větším významem, dochází u všech chaotických dynamických systémů. Tento posuv je tím rychlejší, čím je vyšší největší Lyapunovův exponent $\lambda$. Konkrétně např. pro $\lambda = \ln 2$, což je náš případ zde, bude průměrná rychlost posuvu jedno dvojkové místo na iteraci.

To úzce souvisí s citlivou závislostí na počátečních podmínkách. Malá změna počátečních podmínek se projeví zprvu změnou v zápise čísla někde daleko vpravo za desetinnou čárkou. V průběhu času se tato změna posunuje doleva, tedy blíže k desetinné čárce a nabývá na významu. To je pravá podstata deterministického chaosu. Jeho význam je v tom, že mnoho rozumných matematických modelů reálných fyzikálních a jiných systémů vykazuje právě tento typ chování a je ve velice dobré shodě s naší běžnou i experimentální zkušeností.

V tomto našem případě dynamického systému (3) nastává ta vyjímečná situace, že při posunu informace doleva se číslo zprava doplňuje nulami. Tím po konečném počtu iterací, zde cca 52, což je dáno počtem platných míst mantisy, je celá informace ztracena a nahrazena nulami. Systém pak přejde do stavu s periodou 1, tedy do pevného bodu. Toto je vyjímečná situace, která není typická. V obecném případě, jako např. (1), se většinou zprava nedoplňují nuly, ale číslice, které jsou složitým výsledkem zaokrouhlovacích a jiných chyb.

To se může jevit jako dosti pesimistický důsledek: výsledek simulace chaotického dynamického systému je tedy po malém počtu kroků (zde 52) dán chybami výpočtu a ne fyzikální podstatou studovaného systému.

Skutečnost není tak smutná. Tyto zaokrouhlovací a jiné chyby vlastně dobře simulují termální a jiný šum, přítomný v každém reálném ději. A i v případě reálného systému se malé fluktuace mohou po jisté době projevit makroskopickými změnami chování systému. Důležité je, aby pravá strana dynamického systému dobře popisovala studovaný reálný systém. Pokud tomu tak je, pak růst vlivu malé odchylky bude v reálném systému dobře simulován modelem.

Další dobrá zpráva je stínové lemma (ang. shadowing lemma), které, volně řečeno, zaručuje, že i při nepřesné simulaci, tedy když místo přesné orbity

$$x_{n+1} - f(x_n) = 0$$

počítáme pouze pseudoorbitu

$$|x_{n+1} - f(x_n)| < \delta,$$

existuje taková počáteční podmínka, z které vychází přesná orbita, která je v jistém smyslu blízká naší napočítané pseudoorbitě.

| n | $y_n$ | $y_n$ jak je uloženo v paměti počítače |
|---|---|---|
| 1 | 0,414879 | 00111111 11011010 10001101 01011110 10101100 10000101 11100001 10111000 |
| 2 | 0,829757 | 00111111 11101010 10001101 01011110 10101100 10000101 11100001 10111000 |
| 3 | 0,340486 | 00111111 11010101 11001010 10000101 01001101 11101000 01111001 00100000 |
| 4 | 0,680972 | 00111111 11100101 11001010 10000101 01001101 11101000 01111001 00100000 |
| 5 | 0,638056 | 00111111 11100100 01101010 11110101 01100100 00101111 00001101 11000000 |
| 6 | 0,723887 | 00111111 11100111 00101010 00010101 00110111 10100001 11100100 10000000 |
| 7 | 0,552226 | 00111111 11100001 10101011 11010101 10010000 10111100 00110111 00000000 |
| 8 | 0,895548 | 00111111 11101100 10101000 01010100 11011110 10000111 10010010 00000000 |
| 9 | 0,208903 | 00111111 11001010 10111101 01011001 00001011 11000011 01110000 00000000 |
| 10 | 0,417807 | 00111111 11011010 10111101 01011001 00001011 11000011 01110000 00000000 |
| 11 | 0,835614 | 00111111 11101010 10111101 01011001 00001011 11000011 01110000 00000000 |
| 12 | 0,328773 | 00111111 11010101 00001010 10011011 11010000 11110010 01000000 00000000 |
| 13 | 0,657545 | 00111111 11100101 00001010 10011011 11010000 11110010 01000000 00000000 |
| 14 | 0,684910 | 00111111 11100101 11101010 11001000 01011110 00011011 10000000 00000000 |
| 15 | 0,630180 | 00111111 11100100 00101010 01101111 01000011 11001001 00000000 00000000 |
| 16 | 0,739640 | 00111111 11100111 10101011 00100001 01111000 01101110 00000000 00000000 |
| 17 | 0,520720 | 00111111 11100000 10101001 10111101 00001111 00100100 00000000 00000000 |
| 18 | 0,958560 | 00111111 11101110 10101100 10000101 11100001 10111000 00000000 00000000 |
| 19 | 0,082880 | 00111111 10110101 00110111 10100001 11100100 10000000 00000000 00000000 |
| 20 | 0,165760 | 00111111 11000101 00110111 10100001 11100100 10000000 00000000 00000000 |
| 21 | 0,331521 | 00111111 11010101 00110111 10100001 11100100 10000000 00000000 00000000 |
| 22 | 0,663041 | 00111111 11100101 00110111 10100001 11100100 10000000 00000000 00000000 |
| 23 | 0,673918 | 00111111 11100101 10010000 10111100 00110111 00000000 00000000 00000000 |
| 24 | 0,652164 | 00111111 11100100 11011110 10000111 10010010 00000000 00000000 00000000 |
| 25 | 0,695671 | 00111111 11100110 01000010 11110000 11011100 00000000 00000000 00000000 |
| 26 | 0,608657 | 00111111 11100011 01111010 00011110 01001000 00000000 00000000 00000000 |
| 27 | 0,782686 | 00111111 11101001 00001011 11000011 01110000 00000000 00000000 00000000 |
| 28 | 0,434628 | 00111111 11011011 11010000 11110010 01000000 00000000 00000000 00000000 |
| 29 | 0,869256 | 00111111 11101011 11010000 11110010 01000000 00000000 00000000 00000000 |
| 30 | 0,261488 | 00111111 11010000 10111100 00110111 00000000 00000000 00000000 00000000 |
| 31 | 0,522975 | 00111111 11100000 10111100 00110111 00000000 00000000 00000000 00000000 |
| 32 | 0,954049 | 00111111 11101110 10000111 10010010 00000000 00000000 00000000 00000000 |
| 33 | 0,091902 | 00111111 10110111 10000110 11100000 00000000 00000000 00000000 00000000 |
| 34 | 0,183804 | 00111111 11000111 10000110 11100000 00000000 00000000 00000000 00000000 |
| 35 | 0,367607 | 00111111 11010111 10000110 11100000 00000000 00000000 00000000 00000000 |
| 36 | 0,735214 | 00111111 11100111 10000110 11100000 00000000 00000000 00000000 00000000 |
| 37 | 0,529572 | 00111111 11100000 11110010 01000000 00000000 00000000 00000000 00000000 |
| 38 | 0,940857 | 00111111 11101110 00011011 10000000 00000000 00000000 00000000 00000000 |
| 39 | 0,118286 | 00111111 10111110 01001000 00000000 00000000 00000000 00000000 00000000 |
| 40 | 0,236572 | 00111111 11001110 01001000 00000000 00000000 00000000 00000000 00000000 |
| 41 | 0,473145 | 00111111 11011110 01001000 00000000 00000000 00000000 00000000 00000000 |
| 42 | 0,946289 | 00111111 11101110 01001000 00000000 00000000 00000000 00000000 00000000 |
| 43 | 0,107422 | 00111111 10111011 10000000 00000000 00000000 00000000 00000000 00000000 |
| 44 | 0,214844 | 00111111 11001011 10000000 00000000 00000000 00000000 00000000 00000000 |
| 45 | 0,429688 | 00111111 11011011 10000000 00000000 00000000 00000000 00000000 00000000 |
| 46 | 0,859375 | 00111111 11101011 10000000 00000000 00000000 00000000 00000000 00000000 |
| 47 | 0,281250 | 00111111 11010010 00000000 00000000 00000000 00000000 00000000 00000000 |
| 48 | 0,562500 | 00111111 11100010 00000000 00000000 00000000 00000000 00000000 00000000 |
| 49 | 0,875000 | 00111111 11101100 00000000 00000000 00000000 00000000 00000000 00000000 |
| 50 | 0,250000 | 00111111 11010000 00000000 00000000 00000000 00000000 00000000 00000000 |
| 51 | 0,500000 | 00111111 11100000 00000000 00000000 00000000 00000000 00000000 00000000 |
| 52 | 1,000000 | 00111111 11110000 00000000 00000000 00000000 00000000 00000000 00000000 |
| 53 | 0,000000 | 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 |
| 54 | 0,000000 | 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 |
| 55 | 0,000000 | 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 |

Tabulka 1: Prvních 55 hodnot výsledku simulace dynamického systému (3) na číslicovém počítači zapsáno v desítkové soustavě (druhý sloupec) a zapsáno tak, jak jsou čísla zanesena v paměti počítače (třetí sloupec). Vidíme, jak se zprava šíří nuly až obsadí všechna místa mantisy.