NĚKOLIK POZNÁMEK O GENEROVÁNÍ PSEUDONÁHODNÝCH ČÍSEL Z R(0,1)

Jaromír Antoch, MFF UK, Praha

1. ÚVOD

Význam simulačních metod a metod Monte Carlo v posledním desetiletí nesmírně vzrostl. Důvody jež k tomu vedly jdou od potřeb řešení stále obtížnějších úloh pro něž nejsme schopni nalézt adekvátní analytické řešení, po požadavky časopisů numericky ilustrovat nově navrhované metody. Nezanedbatelným faktorem jsou i možnosti dnešní generace počítačů a pohodlí, s nímž se s nimi pracuje. Současné výpočetní prostředí totiž umožňuje de facto "na stole" realizovat poměrně rychle a snadno i takové projekty, do nichž bychom se dříve raději ani nepouštěli.

Při simulování přitom většina autorů vychází z přesvědčení, že generátor "náhody" který používají je dokonalý. To s sebou implicitně nese víru v to, že i generátor (pseudo) náhodných čísel z R(0,1), sloužící zpravidla za základní kámen "náhody", je dokonalý. Bohužel, každodenní praxe nám ukazuje, že to není zdaleka vždy pravda, spíše naopak.

Cílem tohoto příspěvku je zastavit se především u generátorů (pseudo) náhodných čísel z R(0,1) a zdůraznit jejich úlohu v simulacích. V jednotlivých odstavcích si proto shrneme základní požadavky, jež by měly takovéto generátory splňovat, připomeneme stručně jejich vybrané typy, jakož i základní testy vhodné pro ověření náhodnosti výstupů. V další části se pak zaměříme na vybrané generátory jež se běžně vyskytují okolo nás v překladačích, statistických programových systémech či knihovnách podprogramů, spolu se stručným zhodnocením jejich vlastností. Na závěr si shrneme vybrané závěry týkající se výběru a testování generátorů, resp. vybrané principy a pravidla simulačních metod.

2. TROCHA HISTORIE

Otázka náhody vždy lidi v..rušovala. Například filosofy (a nejen je) k úvahám co to náhoda vůbec je, hazardní hráče (a nejen je) ke snaze o pochopení jejích zákonitostí a možností jejího zahrnutí do svých plánů, a v neposlední řadě též matematiky (a nejen je) ke snaze ji exaktně popsat.

Nicméně již v minulém století si někteří statistici uvědomili, že pro řadu běžných aplikací, jakými je znáhodňování pokusů či výběry z konečných populací, by bylo dobré mít k disposici ať již některé zařízení náhodu - řečeno dnešními slovy-simulující (generující), resp. určitým způsobem sumarizované výsledky takovýchto zařízení. Potřeba opakovatelnosti vedla spíše k druhé alternativě, tj. k vytváření tabulek náhodných číslic a odvozování postupů, jak jich pro potřeby statistiků využívat. Je zajímavé si všimnout, že za tímto účelem lidé po dlouhou dobu užívali především taková zařízení, jež považovali intuitivně za typické zdroje náhody-mince, kostky,mnohostěny, ruletu apod. Později byly za stejným účelem využívány též výsledky censů, viz např. nám všem dobře známé tabulky Kadyrova (1936). Brzy se bohužel ukázalo, že takovéto zdroje zpravidla neposkytují hodnoty "dostatečně náhodné". Bylo to především výsledkem znovu probuzeného zájmu o pojem náhodnosti a odvozování nových testů pro její ověřování.

O něco lepší výsledky poskytla až specielně zkonstruovaná elektronická zařízení, pomocí nichž byly připraveny známé tabulky Kendalla a Babbingtona - Smithe (1939), a především tabulky společnosti RAND (1955), obsahující 10^6 náhodných číslic. Nicméně i v tomto posledním případě bylo nutné výsledky pořízené tzv. elektronickou ruletou upravit pomocí

speciálních metod tak, aby byly odstraněny jisté nepravidelnosti jež se vyskytly. Přes některé nedostatky lze takto připravené tabulky bez problémů používat pro řadu úkolů běžné aplikované statistiky dodnes, např. pro znáhodnění pacientů v klinických pokusech apod. Naproti tomu jsou téměř nepoužitelné i pro jednoduché simulační úlohy, např. už jenom kvůli omezenému rozsahu a nemožnosti eficientně pomocí nich generovat tisíce (desítky tisíc) náhodných čísel potřebné i pro jen jednoduché aplikace.

Potřeba velmi dlouhých sérií "dostatečně náhodných" čísel vedla proto v zásadě dvěma cestami. První z nich byl rozvoj tzv. fyzikálních generátorů, založených většinou na registraci charakteristik určitých fyzikálních pochodů jež mají náhodný charakter. Tyto generátory se příliš neprosadily vzhledem k problému opakovatelnosti.

Druhou cestou byl rozvoj generátorů algoritmických, jimž daly počáteční impuls práce von Neumanna (1946) a Lehmera (1951). Pro širokou obec statistiků a specialistů teorie pravděpodobnosti byl šokem fakt, že deterministické posloupnosti generované jistými algoritmy mohou nahradit náhodné posloupnosti, neboť podle všech tehdy dostupných statistických testů se chovaly jako skutečně náhodné posloupnosti. Poznamenejme, že deterministické posloupnosti nelze rigorózně vzato nazývat náhodnými. A to ani v případě, kdy se jako náhodné chovají. Abychom je proto od skutečně náhodných posloupností odlišili, označujeme je jako pseudonáhodné a odpovídající zdroje zveme generátory pseudonáhodných čísel. Zároveň byla vyvolána nová rozsáhlá vlna filosofických diskusí o pojmu náhodnosti, viz např. Knuth (1981).

Během posledních třiceti let byla, především algoritmickým generátorům, věnována velká pozornost nejen ze strany statistiků, ale především odborníků z oblasti informatiky, numerické matematiky, diskrétní matematiky apod. Jejím výsledkem byly jak stále dokonalejší generátory náhodných čísel - nejen z R(0,1), tak na druhé straně stále lepší pochopení pojmu náhodnosti.

3. GENERÁTORY A SIMULACE

Pravděpodobnostní chování řady statistických postupů často nelze vyšetřovat exaktně. Buď pravděpodobnosti jednotlivých statistik vůbec neumíme spočítat, nebo je tak složité, že je prakticky neupotřebitelné. V podobných situacích máme k dispozici v zásadě dvě možnosti:

- zkoumat asymptotické chování statistik pro velké rozsahy výběru;
- sledovat jejich chování při konečných výběrech pomocí simulací.

V druhém případě jde o to, že pomocí počítače simulujeme řadu konečných výběrů z daného rozdělení pravděpodobnosti a sledujeme hodnoty uvažovaných statistik pro tyto výběry. Nezískáme tak sice analytické vyjádření rozdělení pravděpodobnosti, ale můžeme se k nim numericky přiblížit. Zpravidla tím přesněji, čím více výběrů simulujeme. Neměli bychom přitom zapomínat ani na to, že analytické i simulační postupy jsou spolu velmi úzce svázány. Zatímco analytické prvky tvoří nutný základ pro realizace simulací, simulační metody jsou na druhé straně mnohdy jediným způsobem jak zjistit chování zkoumané procedury.

Vedle již zmíněné aplikaci ve statistice je třeba na druhé straně připomenout, že pro pochopení řady složitých procesů v technice, ekonomii, medicíně, biologii i dalších oborech se stále více a více místo experimentů používá modelování na počítačích. Ještě častěji se v poslední době můžeme setkat s tím, že provádění experimentů s modely na počítačích praktickým pokusům a zkouškám předchází. Aby počítačové experimenty co nejvíce odpovídaly skutečnosti, je třeba do jejich jinak deterministického chodu daného zvoleným modelem vnést prvky náhody - ať již dodatečného šumu, rušení a neočekávaných pozorování, či jiných náhodných dějů běžně se vyskytujících v praxi.

Jak jsme si již uvedli, pokusy generovat hodnoty s náhodnými vlastnostmi s sebou při-

nesly řadu problémů jak teoretických, tak praktických. Mezi ty první patří mimo jiné otázka co to náhoda vůbec je, jak rozhodnout zda daný proces je náhodný či není apod. Je třeba říci, že v literatuře jim byla věnována velká pozornost. Jejím výsledkem jsou mimo jiné následující dva závěry:

- je-li proces generován nějakým algoritmem, potom není náhodný, neboť jej lze za týchž podmínek opětovně zrealizovat;
- jestliže proces je generován nealgoritmicky, to znamená prostřednictvím některého fyzikálního děje, otázka náhodnosti je obecně nerozhodnutelná.

Takže místo otázky, zda námi generovaný proces náhodný je či není, se musíme spíše ptát na to, zda se bude jako náhodný chovat, užijeme-li jej pro svůj problém. Zároveň se ukázalo, že je třeba se zaměřit na vývoj testů, pomocí nichž by bylo možno se utvrdit ve svém přesvědčení o platnosti či neplatnosti výsledků získaných prostřednictvím počítačových experimentů. Připomeňme, že úplně jiný pohled na otázku náhodnosti s sebou přináší především teorie složitosti.

Při simulacích převážná většina autorů vychází z přesvědčení, že má k dispozici DOKONALÝ GENERÁTOR pseudonáhodných čísel z R(0,1), tj. generátor schopný produkovat posloupnosti nezávislých, stejně rozdělených veličin $U_1, U_2, \ldots$ s rovnoměrným rozdělením na intervalu (0,1), aniž se hlouběji zamýšlí, zda tomu tak skutečně je. Bohužel zdaleka ne vždy je to pravda, neboť spíše opak bývá pravdou.

Podíváme-li se, k jakému základnímu účelu statistikovi generátory pseudonáhodných čísel z R(0,1) především slouží, vidíme, že jde o:

- přímé použití,
- generování nezávislých veličin diskrétního typu,
- generování nezávislých veličin spojitého typu,
- generování závislých veličin atd.

a teprve nepřímo skrze ně je ovlivňován i vlastní simulační experiment. To tedy znamená, že v podstatě v každém výskytu náhodné události je zahrnut, ať již viditelně či ne, generátor pseudonáhodných čísel z R(0,1).

Proto by si měl uživatel vždy uvědomovat, že výsledky jeho simulací velice podstatně závisí na kvalitě a rychlost generátorů z R(0,1). To je také hlavní důvod pro to, proč je zapotřebí tolik pozornosti věnovat vývoji, testování a eficientním implementacím těchto "základních stavebních kamenů' všech simulací zahrnujících náhodu. Budeme-li mít k dispozici dobrý generátor, budeme mít mnohem méně problémů s kontrolou generátorů z rozdělení normálního, Poissonova apod. Naopak, bude-li náš generátor defektní, jeho vliv na výsledky simulací bude komplexnější a vliv na jejich kvalitu jenom těžko předvídatelný.

Generování náhodných veličin, ať již diskrétních či spojitých, závislých či nezávislých, náhodných procesů apod., jakož i metodám Monte Carlo, byla v literatuře věnována velká pozornost. Lze říci, že řádově větší než úloze generování nezbytného základu pro ně, tj. generátorům pseudonáhodných čísel z R(0,1). Vzhledem k tomu, že jakékoliv úvahy o této problematice by neúměrnou měrou rozšířily rozsah tohoto příspěvku, nezbývá než odkázat na citovanou literaturu, především pak práce Bratley (1983), Dagpunar (1989), Hurt (1982), Lewis a další (1988), Lewis a Orav (1989) či Walter a Lauber (1975) a odkazy v nich uvedené.

4. ZÁKLADNÍ POŽADAVKY KLADENÉ NA GENERÁTORY

Odborníci i praktici se již dávno shodli na základních vlastnostech, jež by dobrý generátor nezávislých náhodných veličin měl splňovat. Patří mezi ně především následující požadavky:

1) výstup z generátoru by měl co nejlépe aproximovat zvolený typ rozdělení;
2) generátor by měl poskytovat výstup co nejvíce nezávislý, a to i ve vyšších dimenzích;
3) generátor by měl být snadno přenosný mezi počítači různých typů;
4) generátor by měl zaručovat snadnou opakovatelnost výstupu z jednoduše zadaného počátečního bodu;
5) generátor by měl být pokud možno rychlý;
6) generátor by měl mít co možná nejdelší periodu.

Splnit simultánně v praxi tyto zdánlivě jednoduché a přirozené požadavky je však značně náročné, resp. spíše nemožné, a proto se zastavíme u jednotlivých bodů podrobněji. Soustředíme se přitom především na generátory pseudonáhodných čísel z $R(0,1)$.

4.1. Shoda se zvoleným typem rozdělení

Požadavek, aby generátor co nejlépe aproximoval zvolený typ rozdělení je naprosto přirozený a bez jakýchkoliv diskusí. V praxi lze jeho splnění ověřovat především pomocí testů dobré shody a některých testů grafických, viz odstavec 6. Připomeňme, že před každým použitím neznámého generátoru, resp. po každé implementaci nového generátoru, by ověření shody se zvoleným typem rozdělení mělo být vždy jedním z prvních kontrolních testů, který je třeba provést. Vzhledem k tomu, že problematika odstavců 4.1 a 4.2 spolu velmi úzce souvisí, čtenář nalezne některé další poznámky též v následujícím odstavci.

4.2. Nezávislost výstupu

Je zřejmé, že abychom mohli i jen začít se simulacemi obsahujícími náhodu, potřebujeme k tomu jako jeden z nezbytných zdrojů generátor (pseudo) náhodných čísel, tj. generátor realizací náhodných veličin $\{U_i,\ i = 1,2,\ldots\}$, jež jsou nezávislé a mají marginální rozdělení rovnoměrné. Formálně tedy požadujeme, aby pro každou množinu $\{U_{i_1},\ldots,U_{i_k}\}$ platilo

$$P(U_{i_1} \le u_1,\ldots,U_{i_k} \le u_k) = \prod_{j=1}^{k} P(U_{i_j} \le u_j) = \prod_{j=1}^{k} u_j \qquad 0 < u_j < 1, \quad j = 1,\ldots,n ,$$

(4.1)

$$= 0 \qquad \text{jestliže některé } u_j \le 0, \quad j = 1,\ldots,n ,$$

$$= \prod_{\substack{j=1 \\ j \notin J_k}}^{k} u_j \qquad \text{jestliže } u_j > 0, \quad j=1,\ldots,k ,$$

$$\text{a kde } J_k = \{j \mid 1 \le j \le k,\ u_j \ge 1\}.$$

Z (4.1) mimo jiné vyplývá, že zvolíme-li $k = 1$, pak pro všechna i, $i = 1,2,\ldots$ musí platit

$$(4.2) \qquad P(U_i \le u) = F_{U_i}(u) = \begin{cases} 0 & u \le 0 , \\ u & 0 < u < 1 , \\ 1 & u \ge 1 , \end{cases}$$

tzn. marginální rozdělení je rovnoměrné. Podobně, položíme-li $k = 2$, potom musí pro všechny dvojice $\{U_{i_1}, U_{i_2}\}$ platit

$$(4.3)\quad P(U_{i_1} \leq u_1, U_{i_2} \leq u_2) = \begin{cases} 0 & u_1 \leq 0 \text{ nebo } u_2 \leq 0, \\ u_1 u_2 & 0 < u_1, u_2 < 1, \\ u_1 & 0 < u_1 < 1 \text{ a } u_2 \geq 1, \\ u_2 & u_1 \geq 1 \text{ a } 0 < u_2 < 1, \\ 1 & u_1 \geq 1 \text{ a } u_2 \geq 1, \end{cases}$$

atp.

Jelikož vztah (4.3) musí platit pro libovolnou dvojici $\{U_{i_1}, U_{i_2}\}$, jedná se z praktického hlediska o značně silný požadavek na generátor. Například žádá, aby v případě kdy si vykreslíme rozptylový graf realizací dvojic $\{(U_{2i}, U_{2i+1}),\ i = 1,2,\ldots\}$, resp. rozptylový graf realizací dvojic $\{(U_i, U_{i+l}),\ l \in N$ konstanta, $i = 1,2,\ldots\}$, měly by tyto body být rovnoměrně rozděleny v jednotkovém čtverci. Bohužel tomu tak vždy není.

Problém úzce spojený s těmito problémy se týká seriální korelace sousedních, resp. vzdálenějších členů generovaných algoritmickými generátory. Vyčerpávající analýzu tohoto problému lze nalézt v Jansson (1966), stručné shrnutí podává též Hurt (1982).

Dalším požadavkem na generátor je, že by měl být podmíněně nezávislý Ukažme si na příkladě, co tím rozumíme. Tak například podíváme-li se na libovolnou trojici (U_i, U_{i+1}, U_{i+2}), potom by rozdělení vektoru (U_i, U_{i+2}) nemělo dle (4.1) záležet na podmínce kladené na U_{i+1}, řekněme, že U_{i+1} leží v intervalu (0.5, 0.6), nýbrž mělo by být dáno vztahem (4.3). Bohužel,ani v tomto případě to není pravda pro všechny generátory.

Oba předchozí případy jsou jednoduchou ilustrací dobře známého faktu, že některé generátory, především pak kongruenční a Fibonacciho typu, produkují výstup který nevyplňuje $[0,1]^k$, $k \geq 1$, ani skutečně rovnoměrně ani náhodně, nýbrž mají řešetovitou strukturu (lattice structure). Čtenář si může snadno ilustrovat tuto vlastnost např. na generátoru tvaru $V_{n+1} = (5V_n + 1) \bmod (16)$. Zůstaňme na okamžik u kongruenčních generátorů, pro něž lze ukázat, že jejich výstup leží vždy na paralelních nadrovinách. Následující věta udává horní hranici pro počet takovýchto nadrovin.

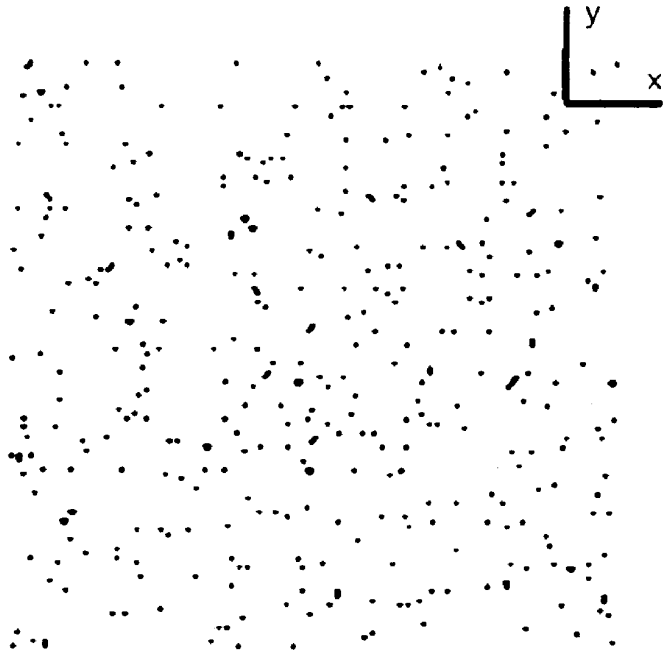
Věta 4.1. Nechť kongruenční generátor tvaru $V_{i+1} = (aV_i + c) \bmod(m)$ produkuje posloupnost hodnot $V_1, V_2, \ldots$ Nechť $U_i = V_i/m$. Potom všechny body tvaru

$$(4.4)\quad \{U_1, U_2, \ldots, U_k\},\ \{U_2, U_3, \ldots, U_{k+1}\},\ \{U_3, U_4, \ldots, U_{k+2}\},\ \ldots$$

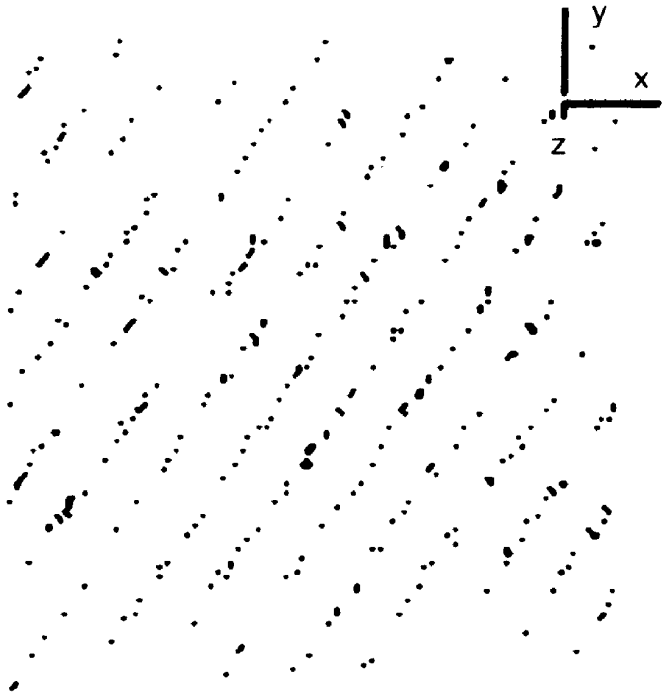
leží v nejvýše $(k!m)^{1/k}$ paralelních nadrovinách. Ilustrujeme si větu 4.1. na dvou jednoduchých případech. Je-li například $m = 2^{32}$ a $k = 10$, potom body tvaru (4.4) leží v nejvýše 41 paralelních nadrovinách, zatímco je-li $m = 2^{32}$ a $k = 2$, body tvaru (4.4) leží v nejvýše 92 681 nadrovinách. První část této ilustrace ukazuje nebezpečí, k jakému může dojít, a zpravidla dochází, při generování nezávislých náhodných vektorů. K výsledku uvedenému v druhé části poznamenejme, že v praxi je nadrovin zpravidla mnohem méně. Například výstupy generátoru RANDU ve dvou dimenzích padají pouze do 15 paralelních nadrovin.

Podíváme-li se z daného hlediska na vlastnosti generátorů doporučované v odstavcích 8. a 9., především pak generátory fi IMSL, Wichmanna a Hilla, LLRANDOMII jakož i generátory doporučované Fishmanem a Moorem, lze říci, že všechny mají z hlediska použití uspokojivou strukturu nejvýše po dimenze 6 – 8.

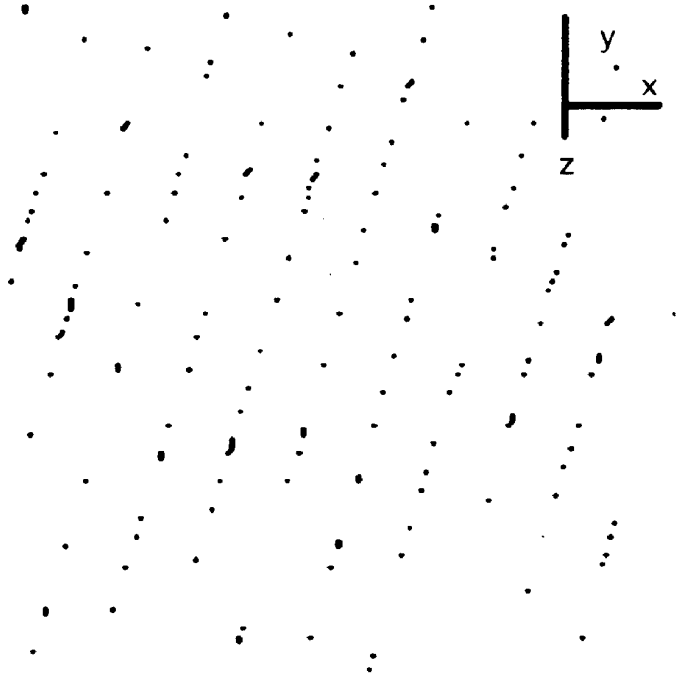
Závěrem učiňme jednu prakticky velmi důležitou poznámku, byť se vztahuje k problematice tohoto odstavce jenom částečně. Jedná se o to, že pseudonáhodná čísla získaná aritmetickými generátory je třeba vždy interpretovat zepředu, pokud možno celá. Jednotlivé cifry, zvláště nižších řádů, nemají vlastnosti náhodných číslic.



Výstup hodnot z množiny $\{(U_i, U_{i+1}, U_{i+2}),\ i=1,\ldots,\ 400\}$ z generátoru RANDU při průmětu do roviny (x,y).



Výstup hodnot z množiny $\{(U_i, U_{i+1}, U_{i+2}),\ i=1,\ldots,\ 400\}$ z generátoru RANDU v projekci znázorněné osovým křížem. Z obrázku je patrné řešetovitá struktura ukazující, že výstup z generátoru RANDU ve třech dimenzích padá pouze do 15 paralelních nadrovin.



Výstup 197 hodnot z množiny $\{(U_i, U_{i+1}, U_{i+2}) \mid U_{i+1} < 0.5),\ i=1,\ldots,400\}$ z generátoru RANDU ukazující opět jeho řešetovitou strukturu, ale i nedobré podmíněné chování výstupu.

4.3. Přenosnost generátorů

Přenosností generátorů rozumíme schopnost generovat pomocí nich přesně týž proud rovnoměrných pseudonáhodných veličin v libovolném čase a na libovolném počítači. Tato problematika je velmi úzce spojena s otázkou, jak generátory produkující tytéž výstupy přenášet mezi jednotlivými počítači s minimem změn a úprav. Je totiž jasné, že u generátoru založeném na posunu bitů budeme potřebovat mnohem více práce při přenosu mezi počítači s různou délkou slova než u jednoduchého kongruenčního generátoru naprogramovaného v některém vyšším jazyku. Přesto i v tomto druhém případě můžeme narazit na řadu problémů, vycházejících z rozličné konstrukce počítačů, překladačů apod. Vzhledem k tomu, že se jedná o velmi složité problémy patřící spíše do problematiky specialistů z oblasti informatiky než statistiky, doporučujeme zájemcům s praktickými dotazy obrátit se na ně. Některé základní informace lze nalézt např. v pracech Schrage (1979) či Gentle (1981).

Pro statistika je tato otázka zvláště důležitá v případě, chceme-li uskutečnit rozsáhlou simulační studii na počítačích různých typů bez obav a starostí o přenosnost a srovnatelnost výsledků. Např. tehdy, naplánujeme-li s více kolegy rozsáhlý simulační experiment, jehož jednotlivé části bude každý z účastníků zpracovávat na svém domovském počítači. V takovém případě je nezbytně nutné, aby jednotliví účastníci věnovali dostatek času testování kompatibility generátorů, jež budou užívat. Jak ukazuje praxe, nestačí se pouze shodnout na jednom algoritmu, nýbrž testovat se musí vždy až konkrétní implementace na jednotlivých počítačích. Jinak totiž může být výsledkem řada značně se lišících generátorů, vycházejících sice z téhož algoritmu, ale produkujících značně rozdílné posloupnosti náhodných veličin s rozdílnými vlastnostmi.

Nechceme-li se otázkou přenositelnosti generátorů příliš zabývat, přestože je přitom nezbytně nutná pro námi plánovanou práci, lze především doporučit použití generátorů firmy IMSL. Tato firma věnovala otázce přenositelnosti velkou péči a jak ukazují testy, její generátory jsou v zásadě přenositelné od IBM PC/XT po CRAY. To se týká nejen generátorů z rovnoměrného rozdělení, ale i z rozdělení normálního, exponenciálního, Binomického, Poissonova a dalších. Pro různé počítače je však třeba užít pro ně připravené speciální implementace.

Otázku přenositelnosti generátorů bychom měli mít alespoň vzdáleně na paměti i z toho důvodu, že počítače, jež ke své práci používáme, se obměňují ve stále se zrychlujícím rytmu. Naproti tomu naše zájmy se zpravidla zdaleka v tak rychlém rytmu nemění a nejen v teorii, ale i v simulacích, je dobré dodržovat "kontinuitu" - např. pro potřeby srovnávání výsledků i po letech.

4.4 Opakovatelnost generátorů

Opakovatelností generátorů rozumíme schopnost generovat přesně týž proud pseudonáhodných veličin v libovolném čase a v libovolné simulaci. Tato otázka je zvláště důležitá v situacích, kdy chceme pomocí simulací srovnávat výsledky dvou či více metod navržených pro řešení téže úlohy, jelikož posouzení rozdílů mezi jednotlivými metodami je mnohem přesnější, jestliže byly použity na tytéž vstupní hodnoty. Opakovatelnost generátoru se též s úspěchem používá u některých speciálních simulačních postupů. Např. při aplikaci metod tzv. redukce rozptylu, umožňujících podstatně snížit počet potřeb simulací při současném zvýšení přesnosti výsledků.

Jinou oblastí, kdy opakovatelnost hraje důležitou roli je situace, kdy simulace produkuje neočekávané výsledky. V takovém případě nás zpravidla zajímá, zda důvodem je "špatný vstup", nebo zda naše očekávání o výsledných hodnotách bylo jednoduše nesprávné. Jsme-li schopni znovu zopakovat původní posloupnost vstupních dat, bude naše situace mnohem snazší. Podobně se s úspěchem využívá opakovatelnost generátorů při ladění programů.

Z hlediska opakovatelnosti je výhodné používat kongruenční generátory a generátory založené na posunu bitů, jež při nastavení těch samých vstupních parametrů produkují vždy

tytéž hodnoty. Samozřejmě na daném počítači a při konkrétní implementaci. Naopak, nemožnost opakovat týž proud pseudonáhodných veličin byla vždy jednou z hlavních námitek proti používání tzv. fyzikálních generátorů, tj. generátorů založených na registraci a vyhodnocování vybraných fyzikálních jevů, například šumu speciálních diod apod. Je proto zajímavé, že v poslední době došlo ve světě, především pak v USA a Japonsku, k značnému oživení zájmu o fyzikální generátory. Důvodem je fakt, že nové optické paměti jsou schopny pojmout desítky TB (tera byte) informace, což umožňuje registrovat značně dlouhé sekvence pseudonáhodných čísel, jež už ze své podstaty nemají řadu záporných vlastností typických pro běžné kongruenční generátory či generátory založené na posunu registrů. Se záznamy na optických nosičích se dále pracuje podobně jako s tabulkami náhodných číslic, tak jako v případě tabulek RAND, Kadyrova apod.

4.5. Výpočetní rychlost

Výpočetní rychlost zajímá každého uživatele z mnoha důvodů. Zpravidla nejen proto, aby příliš dlouho neblokovat počítač a za své výpočty mnoho neplatil, ale především proto, aby zvýšil co nejvíce přesnost svých výsledků. Zvýšíme-li totiž rychlost generátoru pseudonáhodných čísel z R(0,1), můžeme realizovat více simulací a snížit tak mimo jiné rozptyl odhadů závěrečných výstupů ze simulace. Nezanedbatelný je též fakt, že při zvýšení rychlosti generátorů jsme schopni se pustit do simulací složitějších úloh. Je důležité si také uvědomit, že moderní generátory, např. z normálního či gamma rozdělení, lze implementovat v čase pouze 1,3 - 1,5 krát pomalejším než je třeba pro generování rovnoměrně rozdělených pseudonáhodných čísel, takže rychlost těchto generátorů podstatně závisí na rychosti základního generátoru z R(0,1), nikoliv na operacích nutných pro nezbytné transformace na dané rozdělení apod.

Jednou z cest pro zrychlení generátorů je použití asembleru při kódování. Naneštěstí asembler je úzce vázán na použitý typ počítače, takže při jeho užití velmi rychle narazíme na otázku přenositelnosti. Jinou cestou je použití fyzikálních generátorů, schopných pracovat rychostí až 200 kB/sec. Při jejich použití narazíme především na otázku opakovatelnosti, o přenosnosti nemluvme.

Existuje ještě další cesta pro podstatné urychlení práce generátorů pseudonáhodných čísel, bohužel u nás zatím téměř nedostupná. Jedná se o použití paralelních počítačů, resp., v našich podmínkách patrně o poněkud schůdnější způsob spočívající v nasazení přídavných desek umožňujících paralelizaci výpočtů do stávajících počítačů. Relativně jednoduché algoritmy užívané pro generování pseudonáhodných čísel jsou pro takovouto aplikaci velmi vhodné. Zatímco při použití metody "pipe-line" lze dosáhnout podstatného urychlení generátorů pseudonáhodných čísel z R(0,1) počítaných např. některými jednoduchými rekurentními vztahy, při použití maticových procesorů lze dosáhnout nebývalého urychlení při generování náhodných vektorů (matic).

Z hlediska uživatele jde o koncepční změnu zásadního významu. Je přitom zřejmé, že při případném použití jde navíc o velký krok do neznáma. Před (potenciálním) uživatelem se zákonitě objeví řada nových, z větší části jen nedokonale zmapovaných problémů, např. :

- jak paralelizovat svůj vlastní simulační problém ;
- jak klasické algoritmy pro generování pseudonáhodných čísel paralelizovat při tom kterém typu paralelního zpracování ;
- jak ověřit náhodnost vystupujících hodnot ;
- v neposlední řadě hlavním problémem asi bude potřeba zásadní změny svého myšlení a přístupu k algoritmizaci problémů.

4.6. Délka cyklu

Jak je všeobecně známo: jednou ze základních špatných vlastností algoritmických generátorů pseudonáhodných čísel je ta skutečnost, že jsou schopny produkovat bez opakování pouze posloupnosti konečné délky. I u nejlepších z nich to znamená, že vždy po určité době jejich chodu dojde k opakování, a to ať již celé doposud nagenerované posloupnosti, resp. její části. Mezi přirozené požadavky většiny uživatelů samozřejmě patří, aby k opakování docházelo pokud možno po co nejdelší době. Důvodů k tomu je celá řada. Například pro některé typy simulací v teoretické fyzice či geologii je třeba mít k disposici okolo 2^{50}-2^{80} různých hodnot, což je mnohem více než umožňují běžně užívané generátory. Dále chceme, aby vzhledem k řešetovitému charakteru výstupu generátorů, především kongruenčního typu, nagenerované hodnoty co nejlépe vyplňovaly interval (0,1), resp. vektory tvaru (4.4) aby co nejlépe vyplňovali $(0,1)^k$.

Tyto požadavky vedly řadu matematiků ke studiu podmínek, jež by jednotlivé generátory měly splňovat, aby maximálně možná délka jejich cyklu byla co nejdelší. Ještě důležitějším úkolem daného typu bylo, jak konkrétně navrhnout generátor, aby reálná délka cyklu při libovolné volbě počátečních podmínek dosahovala vždy maximálně možné hodnoty. Konkrétní výsledky a doporučení shrnuje např. Knuth (1981). Mezi algoritmické generátory s velmi dlouhým cyklem patří generátory založené na posunu registrů, kde byl například navržen a podrobně studován generátor s astronomickou periodou 2^{607}-1. Jiná cesta vedoucí ke zvyšování délky cyklu algoritmických generátorů vede přes jejich kombinování.

5. VYBRANÉ TYPY GENERÁTORŮ

Pro potřeby generování rovnoměrných pseudonáhodných čísel máme v praxi k disposici v podstatě dvě základní možnosti, tj. použití fyzikálních nebo algoritmických generátorů. Jak uvidíme, každá z nich má své přirozené přednosti i zápory, a volba mezi nimi záleží na našich potřebách.

Vývoji a testování generátorů rovnoměrných pseudonáhodných čísel byly v uplynulých letech věnovány stovky prací. Výsledkem tohoto společného úsilí je skutečnost, že v současné době máme k disposici velký výběr poměrně dostupných generátorů, řadu z nich s velmi dobrými vlastnostmi. Cesta k nim však nebyla v žádném případě jednoduchá a přímá, jak by se mohlo zdát z několika formulek je definujících či poznámek k nim připojených. Naopak, byla značně trnitá. Cílem tohoto odstavce je připomenout si stručně původ a základní vlastnosti generátorů, jakož i upozornit na některé problémy, s nimiž se můžeme při jejich používání setkat.

5.1. Fyzikální generátory

Vedle algoritmických generátorů se v praxi používají, byť ne zdaleka tak často, též fyzikální generátory. Jejich podstata zpravidla spočívá v registraci charakteristik určitých pochodů majících náhodný charakter, např. rozpadu radioaktivních látek, šumu ve výbojkových trubicích či u speciálně navržených diod apod. Uvedená zařízení přitom mohou být buď součástí počítače, nebo pracovat jako externí zařízení a být pouze připojena přes některý vstup.

Mezi výhody fyzikálních generátorů patří především rychlost až 200kb/sec, resp. dobrý psychický pocit uživatele vycházející z důvěry v náhodnost užitého procesu, jehož realizace registrujeme. Mezi základní nevýhody patří naopak problém opakovatelnosti, podrobně diskutovaný v odstavci 4. Dále je to potřeba neustálé kontroly, zda nedošlo ke změně u generujícího fyzikálního procesu, např. skoku v úrovni apod. Nicméně, jak jsme se již zmínili dříve, fyzikální generátory prožívají v poslední době jistou renesanci díky velké kapacitě optických disků umožňujících uschovat až desítky TB informace. Studium fyzikálních generátorů se tak v současné době soustředilo především na otázky typu:

- jak kontrolovat náhodnost takto získaných hodnot;

- metody promíchávání získaných hodnot za účelem zvýšení jejich náhodnosti podobně jako tomu bylo u tabulek RAND;
- konstrukci nových typů, jež by byly schopné samokontroly kvality výstupu dle zadaných požadavků;
- v neposlední řadě též na to, aby byly skutečně efektivně využívány.

Poznamenejme, že tento poslední požadavek by se měl stát jednou ze základních zásad při požívání generátorů libovolného typu.

5.2. Von Neumanův generátor

Von Neumannova metoda tzv. prostředních řádů druhé mocniny, navržená již v roce 1946, se stala předchůdcem moderních algoritmických postupů. Princip metody je velmi jednoduchý:

1) zvolí se vhodné počáteční číslo seed a zobrazí se v k dvojkových místech, tj. binárně;
2) spočte se $seed^2$ a uloží se 2k dvojkových místech;
3) prostředních k bitů z této druhé mocniny se vezme za další člen posloupnosti a celý postup se opakuje.

Poznamenejme, že není samozřejmě nutné ukládat číslo pouze binárně, proceduru lze realizovat i dekadicky.

Přestože tato metoda má cenu pouze historickou, neboť nesplňuje současné požadavky kladené na dobré generátory, její přínos ve své době byl značný. Nejenže na ní byly ukázány základní obecné principy a požadavky jež by algoritmické generátory měly splňovat, ale především podstatně změnila nazírání lidí na otázku generování náhody. Von Neumann se totiž především zaměřil na následující problémy:

- ukázal, že výstup z jeho generátoru se navzdory deterministickému principu generování může chovat velmi náhodným způsobem;
- požadoval, že je třeba znát co nejvíce teoretických výsledků o generátorech, např. o délce jejich cyklu, struktuře výstupu, korelovanosti členů výstupní posloupnosti, nezávislosti atp. před tím, než se budou používat;
- požadoval, že teoreticky podložená náhodnost, stejně tak jako i ostatní vlastnosti generátorů, musí být verifikovány empiricky pomocí baterie testů, jež je třeba za tím účelem vyvíjet.

5.3. Kongruenční generátory

Nejvíce používanou a nejlépe prostudovanou třídou algoritmických generátorů jsou tzv. kongruenční generátory zavedené Lehmerem (1951). Ten navrhl pro generování používat lineární kongruenční vztah

$$V_i = (a\,V_{i-1} + c) \bmod (m), \qquad i = 1,2,\ldots, \tag{5.1}$$

kde V_0 je některá počáteční startovací hodnota (seed) a $\{a,c,m\}$ jsou předem vhodně zvolené pevné konstanty. Abychom dostali číslo z R(0,1), je třeba vracet místo V_i hodnoty $U_i = V_i/m$, $i = 1,2,\ldots$ Položíme-li ve vztahu (5.1) $c=0$, hovoříme o multiplikativním kongruenčním generátoru, zatímco je-li $c \neq 0$, jedná se o smíšený kongruenční generátor.

Obecněji se za kongruenční považují ty generátory, jež jsou založeny na rekurentním vztahu

$$V_i = (a_0\,V_{i-1} + \ldots + a_{k-1}\,V_{i-k} + c) \bmod (m), \qquad i=k,\ k+1,\ldots, \tag{5.2}$$

kde $V_0,\ldots,V_{k-1}$ jsou některé počáteční startovací hodnoty (seeds) a $\{a_0,\ldots,a_{k-1},c,m\}$ jsou předem vhodně zvolené pevné konstanty. Položíme-li v (5.2) $a_0=\ldots=a_{k-1}=1$, hovoříme o aditivním kongruenčním generátoru. Nejznámějšími generátory tohoto typu jsou tzv. Fibonacciho generátory dané vztahem

(5.3) $$V_i = (V_{i-1} + V_{i-2}) \bmod (m),$$

o nichž se blíže zmíníme v odstavci 5.3.

Všimněme si nejprve některých základních vlastností těchto generátorů.

1. Vzhledem k užité operaci modulo je zřejmé, že výsledkem jsou čísla z množiny $\{0,1,\ldots,m-1\}$. Zajímáme-li se tedy o rovnoměrná čísla z R(0,1), nejjemnější dělení jež generátor může poskytovat je množina $\left\{0,\frac{1}{m},\ldots,\frac{m-1}{m}\right\}$. Toto není samozřejmě skutečně rovnoměrné rozdělení na (0,1). Na druhé straně se jedná o typický problém, s nímiž se setkáváme u všech počítačových algoritmů užívaných pro generování pseudonáhodných čísel.

2. Jelikož hodnota V_i záleží pouze na předchozí hodnotě V_{i-1} a pevně daných konstantách $\{a,c,m\}$, poté co se jedna hodnota opakuje, musí se opakovat i celá posloupnost hodnot následujících po ní. Jak již víme, takováto posloupnost se nazývá cyklus a jeho perioda délkou cyklu. Je zřejmé, že maximální délka cyklu kongruenčních generátorů je rovna m. Na druhé straně existují příklady kongruenčních generátorů jež obsahují mnoho krátkých cyklů a ten, do nějž vstoupíme, záleží pouze na počáteční hodnotě V_0. V literatuře byla publikována celá řada výsledků ukazujících jak volit konstanty $\{a,c\}$ tak, abychom při dané hodnotě m dosáhli maximální délky cyklu. Shrnutí viz Knuth (1981), řadu výsledků uvádí též Hurt (1982).

3. Kongruenční generátory mají řešetovitou strukturu. Volba konstant $\{a,c,m\}$, stejně jako zvolená aritmetika v níž výpočty provádíme, určuje nejenom jemnost dělení intervalu (0,1) a délku cyklu, ale též rovnoměrnost marginálních rozdělení a nezávislost výstupní posloupnosti. Vhodná volba parametrů definujících generátor je "uměním" kombinujícím jak teoretické výsledky, tak výsledky empirických testů. O některých vhodných volbách se lze více dozvědět v odstavích 8. a 9. Poznamenejme, že z teoretického hlediska je značný rozdíl mezi smíšenými a multiplikativními generátory, vedoucí na dosti zásadně různá doporučení pro volbu konstant $\{a,m\}$.

4. Pro multiplikativní generátory s prvočíselným základem m je charakteristické, že dolní bity v jejich reprezentaci jsou vysoce nenáhodné. Situace je přitom tím horší, čím nižší bity nás zajímají. Zvláště patrná je tato vlastnost u generátorů počítaných modulo 2^k. Na druhé straně je zajímavé si všimnout, že multiplikativní generátory s prvočíselným modulem mající plný cyklus vykazují poměrně dobré chování dolních bitů, ačkoliv neexistuje žádná teorie schopná tento jev vysvětlit plné spokojenosti.

Zbývá tedy základní otázka, totiž, jak zvolit optimálně hodnoty konstant $\{a,c,m\}$, tj. jaký konkrétní generátor používat. Nejsme-li profesionálními návrháři generátorů pseudonáhodných čísel z R(0,1), můžeme postupovat například následovně:

a) Vybrat si na základě literatury některý z všeobecně doporučovaných generátorů a implementovat jej na svém počítači.
b) Danou implementaci podrobit baterii testů a výsledky řádně zdokumentovat. Některé vhodné testy viz odstavec 6., řadu dalších uvádí citovaná literatura.

c) Nezapomínat na to, že žádný generátor není absolutně dokonalý. Vždy je třeba si vybrat takový typ, který nejvíce vyhovuje naší aplikaci, tj. jehož vady jsou z hlediska konkrétního použití nejvíce zanedbatelné.

5.3. Fibonacciho generátory

Další třídu algoritmických generátorů o nichž se zmíníme tvoří tzv. Fibonacciho generátory, jejichž jméno pochází ze známé Fibonacciho rekurentní formule. Jedná se o speciální typ obecného lineárního kongruenčního generátoru, v němž nová hodnota je počítána přímo ze dvou předchozích vztahem

$$(5.4) \qquad V_i = (V_{i-1} + V_{i-2}) \bmod (m) .$$

Poněkud obecnější definice generátorů Fibonacciho typu vychází ze vztahu

$$(5.5) \qquad V_i = V_{i-r} \otimes V_{i-s}, \qquad r \geqslant 1, \quad s \geqslant 1, \quad r \neq s,$$

kde symbol $\otimes$ označuje libovolnou vhodnou matematickou operaci. Přitom v (5.2) V_i mohou být jak binární vektory, tak celá čísla či reálná čísla z (0,1) apod., záleží pouze na volbě operace $\otimes$. Tedy například:

- V_i jsou reálná čísla a $\otimes$ representuje sčítání nebo odčítání modulo 1;
- V_i jsou (n-1)-bitová celá čísla a $\otimes$ representuje sčítání, odčítání či násobení modulo 2^n;
- V_i jsou binární vektory a $\otimes$ representuje binární sčítání, odčítání nebo násobení, resp. binární operací exclusive-or apod.

Výhodou těchto generátorů je jejich rychlost. Mezi hlavní nevýhody patří krátká délka cyklu a značná závislost kvality generátoru na volbě počátečních (startovacích) hodnot. Knuth (1981) navíc ukázal, že struktura generátoru může být, zvláště ve vyšších dimenzích, značně neuspokojivá. Na druhé straně jistého vylepšení lze dosáhnout pomocí míchání.

Na závěr tohoto odstavce uveďme, že po vyjití práce Marsaglia (1985) byl v literatuře opětovně probuzen zájem o studium generátorů Fibonacciho typu. Existuje tak jistá reálná možnost, že v dohledné budoucnosti se setkáme s návrhy kvalitních generátorů Fibonacciho typu směle soupeřících svou kvalitou s nejlepšími representanty tříd generátorů ostatních.

5.4. Tauseworthův generátor

Tento generátor, navržený Tauseworthem (1965), je v literatuře znám spíše pod názvem generátor založený na posuvu registrů, a v praxi se příliš často nepoužívá. Nicméně některé jeho přednosti nás vedou k tomu, aby jsme se u něj na okamžik zastavili. Vlastní konstrukci si přiblížíme na jednoduchém příkladu generování celých čísel z intervalu $[0,127]$ representovaných sedmibitovými binárními vektory, pomocí " posunů o dva bity vpravo a tři bity vlevo " za použití operace exclusive-or.

Užitá operace	Výsledek operace
1) Zvolme některý sedmibitový vektor jako počáteční hodnotu (různý od (0,0,0,0,0,0,0)).	(1,0,0,1,1,0,1)
2) Proveďme posuv o dva bity vpravo.	(0,0,1,0,0,1,1)
3) Proveďme operaci exclusive-or na vektory z bodů 1) a 2).	(1,0,1,1,1,1,0)
4) Proveďme posuv o tři bity vlevo.	(1,1,1,0,0,0,0)
5) Proveďme operaci exclusive-or na vektory z bodů 3) a 4).	(0,1,0,1,1,1,0)

Výsledná hodnota (0,1,0,1,1,1,0) je naším prvním nagenerovaným náhodným číslem, jež lze dále použít v bodě 1). Připomeňme, že operace exclusive-or je definována vztahy 0+0=1+1=0, resp. 0+1=1+0=1.

Posun o bit vpravo, resp. vlevo, lze jednoduše representovat pomocí binární matice $\underset{\sim}{R}$, resp. $\underset{\sim}{L}$, což umožňuje charakterizovat Tauseworthův generátor některou binární maticí $\underset{\sim}{I}$, jíž postupně násobíme počáteční binární vektor $\underset{\sim}{b}$. Výstupem pak je posloupnost hodnot

(5.6) $$\{ \underset{\sim}{b}, \underset{\sim}{b}\underset{\sim}{I}, \underset{\sim}{b}\underset{\sim}{I}^2, \ldots \} .$$

V našem příkladě $\underset{\sim}{I}=(1+\underset{\sim}{R}^2).(1+\underset{\sim}{L}^3)$. Poznamenejme, že existují i jiné způsoby representace těchto generátorů.

Mezi výhody Tauseworthova generátoru mimo jiné patří :

- možnost dosáhnout astronomické délky cyklu, např. s použitím výsledků Zierlera (1969) až $2^{607}-1$;
- při použití asembleru jsou výsledkem elegantní a velmi rychle pracující implementace ;
- vícerozměrné chování je velmi dobré zvláště u implementací na počítačích s krátkým slovem, v tomto případě mnohdy předčí analogické chování jednoduchých kongruenčních generátorů.

Mezi nevýhody Tauseworthova generátoru naopak patří :

- řešetovitá struktura ;
- především pak nedostatek dobře rozpracované teorie, popisující komplexně jejich vlastnosti a nabízející vhodná doporučení pro volbu " optimálních posuvů ".

Jak jsme viděli, tento generátor má tři vstupní parametry. Některou počáteční hodnotu jež musí být různá od nuly a dvě konstanty definující počet posuvů vpravo, resp. vlevo. Vzhledem k již zmíněnému nedostatku vhodné teorie je třeba závěrem říci, že patrně nejbezpečnější strategií při výběru generátoru Tauseworthova typu je následující – použít hodnoty doporučované v literatuře, např. některý z příkladů navržený Toothilem (1971, 1973) apod.

5.6. Kombinování generátorů

Jak jsme viděli, každý z algoritmických generátorů jež jsme doposud popsali má některé záporné vlastnosti. Jednou z cest, jak se matematici snažili tyto nedostatky odstranit, se staly různé způsoby kombinování generátorů. V prvopočátku se jednalo o pokus potvrdit intuitivní víru v to, že tak lze zlepšit jak náhodnost, tak rovnoměrnost výstupu. Některé teoretické výsledky, viz např. Rosenblat (1975), Brown a Solomon (1979) či Marsaglia (1985), tuto víru později potvrdily. Nešlo však pouze o náhodnost a rovnoměrnost. Neméně důležitý byl i fakt, že se ve všech případech podařilo výrazně prodloužit délku cyklu kombinovaného generátoru oproti délce cyklů generátorů jej tvořících. Toto byl velmi důležitý důsledek zvláště v případě generátorů implementovaných na počítačích s krátkou délkou slova.

Jednou z cest kombinování generátorů je použít dva nebo tři, mohou mít i poměrně krátký cyklus, a postupovat podobně jako u generátoru Wichmanna a Hilla, viz odstavec 9. Častějším způsobem je ale použití tzv. míchání (shuffling). Popišme si stručně formou algoritmu verzi, jež se patrně nejčastěji používá a jíž pod názvem metoda smíšených generátorů navrhli MacLaren a Marsaglia (1965). Pro jeho použití potřebujeme mít k dispozici generátor pseudonáhodných čísel z R(0,1) produkující hodnoty $Y_1, Y_2, \ldots$, a generátor pseudonáhodných čísel $R_1, R_2, \ldots$ z diskrétního rovnoměrného rozdělení z $\{1,2,\ldots,n\}$.

Algoritmus 5.1.
Nageneruj vektor $\underset{\sim}{Y} = (Y_1, \ldots, Y_n)$ z R(0,1).
i:=0
REPEAT
    i:=i+1
    Nageneruj R_i z $\{1,2,\ldots,n\}$.
    Nageneruj Z z R(0,1).
    $U_i := Y_{R_i}$: $Y_{R_i} := Z$
    UNTIL Nageneroval jsem již dostatek hodnot.
RETURN $U_1, U_2, \ldots$

Praktické testy ukázaly, že tento generátor poskytuje zpravidla lepší výsledky než generátory, z nichž je složen. Pouze v případě, kdy jsou oba generátory velmi špatné, může selhat. O jiných způsobech míchání se lze dočíst v citované literatuře.

6. TESTOVÁNÍ GENERATORŮ NÁHODNÝCH ČÍSEL

Jak jsme již mnohokrát zdůraznili, před použitím neznámých generátorů, resp. po každé vlastní implementaci generátoru doporučovaného v literatuře, je třeba vždy provést alespoň základní otestování jejich vlastností. Ověříme si tak, zda výstup který generátor poskytuje se chová jako náhodný výběr z požadovaného rozdělení. Zpravidla přitom nestačí užít jeden nebo dva testy, nýbrž je jich třeba aplikovat celou baterii.

Je samozřejmé, že naší snahou by vždy mělo být používání co možná nejlepšího generátoru. Jelikož na druhé straně je téměř nemožné u jednoho generátoru dosáhnout špičkových výsledků ve všech sledovaných parametrech, lze doporučit mít implementováno vždy generátorů několik a vybírat z nich podle typu úlohy, jíž chceme řešit. Používání generátorů se známými kvalitami i vadami je totiž vždy mnohem bezpečnější než užívání neznámých generátorů "naslepo".

Testů ověřujících kvalitu generátorů byla v literatuře navržena celá řada. Od klasických frekvenčních u nichž nezáleží na pořadí generovaných hodnot, přes testy náhodnosti u nichž na pořadí generovaných hodnot podstatně záleží, až po testy grafické. Důležité je též si uvědomit, že zatímco řada testů je globálního charakteru, neméně důležité jsou i testy potřebné k ověření lokální nezávislosti.Každý z testů přitom ověřuje vždy jen splnění určitého požadavku kladeného na generátor, a to na jednotlivém (několika málo) výstupu generátoru konečné délky, jež může být velmi závislý na počátečních hodnotách. Jelikož převážná většina testů byla podrobně popsána i v české literatuře, viz např. Hurt (1982) či Walter a Lauber (1975), nebudeme se jimi podrobně zabývat, nýbrž si je pouze stručně připomeneme.

a) Frekvenční testy slouží pro ověření rovnoměrnosti rozdělení generovaných čísel. Zpravidla se užívá χ^2-test dobré shody nebo testy Kolmogorovova-Smirnovova typu. K podobnému účelu slouží též test založený na hodnotách součtů po sobě jdoucích m-tic generovaných veličin, $m \geq 5$.

b) Jak jsme viděli v odstavci 4., velmi důležité je též ověření toho, jak dobře generované hodnoty vyplňují interval $(0,1)^k$, z níž mimo jiné můžeme usuzovat na nezávislost generátoru ve vyšších dimenzích. Pro $k=2$ se jedná o tzv. seriální test, ověřující pomocí χ^2-testu dobré shody zda posloupnost $\{(U_{2i}, U_{2i+1}),\ i=0,1,\ldots\}$ je rovnoměrně rozdělena v jednotkovém čtverci. Pro $k > 2$ se test konstruuje analogicky

c) Testy řešetovistosti se provádějí především pro generátory kongruenční. Verze pro $k = 2$, nazývaná Gruenbergův test, je podrobně popsána v Hurt (1982), její zobecněné pro $k > 2$ uvádí Atkinson (1980). Viz též práce Marsagliovi.

d) Test maxima , resp. minima, slouží pro ověření vhodnosti generátorů například pro simulace chvostů rozdělení, extrémálních pořádkových statistik apod. Test spočívá v tom, že generujeme určitý počet posloupností pevné délky a zjišťujeme pomocí frekvenčního testu, zda rozdělení maxima, resp. minima, v těchto podposloupnostech souhlasí s očekávaným teoretickým rozdělením.

e) Poker test slouží k ověření lokální náhodnosti.Výstupy z generátoru se užívají k formování figur analogických pokeru a testuje se shoda relativních četností výskytů těchto figur s očekávaným teoretickým rozdělením. K podobnému účelu slouží též test náhodnosti výskytu číslic, který spočívá v tom, že se zvolí určitá číslice a spočtou se relativní četnosti výskytu "mezer" mezi opětovnými výskyty této číslice. Některým z testů dobré shody se dále otestuje, zda se relativní četnosti shodují s teoreticky spočtenými pravděpodobnostmi. Vzhledem k jeho charakteru se mu říká test mezer, a lze jej snadno upravit i pro případ testování náhodnosti výskytu generovaných čísel v intervalu (0,1).

f) Provádění seriálního testu může být, zvláště ve vyšších dimenzích, značně náročné na čas i na paměť počítače.Proto je někdy snazší podívat se na celý problém z druhé stránky a užít raději tzv. test kolizí, jež tvoří jeho protipól. Princip tohoto testu spočívá v tom, že místo abychom registrovali počty bodů jež padly do jednotlivých buněk rozkladu intervalu $(0,1)^k$, budeme zaznamenávat pouze počet buněk do nichž padl více než jeden bod, resp. alespoň dva body apod. Tento přístup vede na velmi pěkné a zajímavé pravděpodobnostní úlohy analogické problémům známým ze statistické fyziky (Maxwellova-Bolzmanova statistika apod.). Z hlediska implementace se přitom nabízí řada elegantních a na paměť úsporných řešení, např. pomocí polí bitů apod.Podrobnější úvahy lze nalézt např. v práci Dagpunar (1989).

g) Vybrané neparametrické testy náhodnosti. Předpokládejme, že námi generovanou posloupnost tvoří realizace nezávislých, stejně rozdělených náhodných veličin U_i, $i=0,1,2,\ldots,n$. Obáváme-li se, že tato posloupnost obsahuje některé systematické změny hodnot při rostoucím indexu i, lze se o platnosti této hypotézy přesvědčit pomocí řady testů. Důležitou roli mezi nimi hrají testy neparametrické. Vzhledem k jejich důležitosti si připomeňme několik nejtypičtějších z nich jež jsou vhodné pro dané účely. Při výběru mezi nimi je přitom třeba mít vždy na paměti především tvar alternativní hypotézy a se zřetelem k ní vybírat vždy vhodný test mající největší sílu.

Obáváme-li se výskytu trendu, speciálně trendu lineárního, lze užít některý z běžně známých postupů, např. Spearmanův korelační koeficient aplikovaný na dvojice (i,U_i), $i=0,1,2,\ldots,n$, test podle znamének diferencí (známý též jako test založený na počtu bodů růstu)či některý z testů iterací. Dochází-li však ke změnám které mají spíše periodický charakter, je třeba užít jiná kritéria. Důvodem je fakt, že předchozí testy mohou úplně selhat proti alternativě symetrických oscilací. V takovém případě lze spíše doporučit použití testu založeného na bodech zvratu, tj. de facto na počtu výskytů lokálních minim a maxim v nagenerované posloupnosti. Podrobný popis lze nalézt např. v učebnici Anděl (1978).

h) Velice důležitým požadavkem kladeným na posloupnost generovaných náhodných čísel je nezávislost po sobě jdoucích členů. Je přitom ale zřejmé, že již díky genezi převážné většiny námi uvažovaných generátorů je tato vlastnost u nich zpravidla více či méně narušena. Tak například u kongruenčních generátorů je V_{i+k} jednoznačně určeno hodnotou V_i apod. Na druhé straně je jasné, že při konstrukci, resp. volbě generátorů bychom měli v každém případě dbát co nejvíce na to, aby závislost byla co možná nejmenší, a to nejen u bezprostředně následujících, ale i vzdálených veličin.

Uvedený problém má samozřejmě dvě stránky, teoretickou a praktickou. Je třeba přitom říci, že z analytického hlediska přináší studium korelace sousedních, resp. vzdálenějších, členů poměrně značné potíže. Pro zajímavost si uveďme, že pro kongruenční generátory tvaru $V_{i+1}=(aV_i+c)\bmod(m)$ lze ukázat, že

$$\varrho_1 = \varrho_1(V_i,V_{i+1}) \approx a^{-1} + 6c^2a^{-1}m^{-2} - 6ca^{-1}m^{-1} \pm am^{-1} .$$

Podrobnosti, stejně jako výsledky pro ϱ_k, $k \geq 1$, lze nalézt např. v práci Jansson (1966).

Poznamenejme, že zatímco pro některé kombinace hodnot (a,c,m) jsou tyto aproximace dobré, pro jiné mohou být naprosto nevyhovující. Bohužel, zpravidla nám ani příliš nepomohou při hledání optimálních či alespoň suboptimálních konstant definujících generátor, takže se jedná spíše o způsob získání globální představy o situaci.

V praxi většinou slouží pro odhalení závislostí mezi po sobě následujícími členy všem dobře známé výběrové koeficienty seriální korelace r_k. Jelikož lze poměrně snadno dokázat, že jsou asymptoticky normálně rozdělené, lze významnost korelovanosti po sobě jdoucích členů poměrně jednoduše testovat. Na závěr poznamenejme, že testy tohoto druhu je třeba u každého nového generátoru vždy pečlivě provést, neboť se jedná o jedno z nejdůležitějších ověření kvality generátoru. Přitom je známo, že právě výsledky těchto testů jsou jenom málokdy k dispozici, resp. zveřejňovány. Bohužel,

7. NĚKTERÉ PROBLÉMY VÝPOČETNÍHO RÁZU

Jak je uvedeno na více místech tohoto příspěvku, existuje celá řada dobrých algoritmů pro generování náhodných čísel poskytujících, alespoň teoreticky, dostatečně náhodný výstup. Přesto tyto algoritmy nemusí v praxi pracovat ani optimálně, ani nemusí poskytovat na různých počítačích, resp. na různých pracovištích a těch samých počítačích, tytéž hodnoty. Důvodů k tomu je celá řada. Zpravidla jsou více či méně úzce svázány s jednou z následujících skutečností :

a) mezi algoritmem a jeho reálnou implementací bývá, mnohdy značný, rozdíl ;
b) počítač na kterém pracujeme nemusí být schopen vyčíslit daný algoritmus absolutně správně ;
c) triky, které používáme ať již ke " zkvalitnění " či ke zrychlení práce generátoru, mohou být zavádějící.

Vedle toho se běžně setkáváme s řadou potenciálních, více či méně skrytých problémů, jež se sice na první pohled mohou zdát banálními, ale které ve svých důsledcích občas přinášejí nečekaná rozčarování a zklamání. Podívejme se proto blíže na dva nejčastěji se vyskytující typy potíží.Naši pozornost sice soustředíme na kongruenční generátory, ale je zřejmé, že s naprosto analogickými problémy se můžeme setkat i u ostatních typů generátorů.

7.1. Vliv užité aritmetiky

Uvažujme jednoduchý kongruenční multiplikativní generátor LLRANDOM I tvaru $V_{i+1}=(16807V_i) \mod(2^{31}-1)$. Fakt, že všechna V_i jsou z intervalu $(0,2^{31}-1)$ by mohl svádět k použití 32-bitové celočíselné aritmetiky především za účelem zrychlení generování. Uvědomíme-li si však, že $16807.(2^{31}-1) > 2^{32}$, vidíme, že mnohem správnější by měla být implementace založená na 64-bitové reálné aritmetice v dvojité přesnosti. Podobně generátor Wichmanna a Hilla, podrobně studovaný v odstavci 9., svádí k použití 16-bitové celočíselné aritmetiky, ačkoliv při jejím neopatrném použití se můžeme snadno setkat s podobnými potížemi.

Na druhé straně je třeba říci, že oba tyto generátory lze též naprogramovat při použití pouze celých čísel, viz např Schrage(1979) či Wichmann a Hill (1982,1984). Vlastní implementace však není vůbec přímočará, naopak, oba algoritmy jsou v tomto případě dosti složité a výsledná rychlost není o mnoho vyšší než při přímém použití 64-bitové reálné aritmetiky v dvojité přesnosti. Fakt, který může být značně překvapující pro řadu uživatelů.

Další nečekané problémy a překvapení s sebou může přinést též to, kolik bitů reálně používá ten který procesor (koprocesor) pro representaci čísel. Potíže s tím spojené popisuje na příkladu implementace algoritmu Wichmanna a Hilla na počítači Prime-400 McLoad (1985). Jak se ukázalo, základní příčinou jeho problémů byl fakt, že tyto počítače užívají pouze 23 bitů pro representaci mantisy.

Na tomto místě je třeba si otevřeně přiznat, že podobnými otázkami se v praxi jen málokdo skutečně zabývá. Nemluvě o tom, že by běžný uživatel podrobně zkoumal, jak přesně pracuje ten který překladač na právě jeho počítači. Bohužel přitom právě toto může být oním zdrojem řady " jen těžko vysvětlitelných chyb ".

7.2. Přesnost výpočtu

Uvažujme kongruenční generátor typu $V_{i+1}=(950\ 706\ 376.V_i) \mod(2^{31}-1)$ a naprogramujme jej podle této formule v IBM Professional FORTRANu na IBM PC/AT s koprocesorem v 64-bitové reálné aritmetice v dvojité přesnosti. Poznamenejme, že v takovém případě by nemělo docházet k žádným problémům s přeplněním paměti apod. Problémy však začnou v okamžiku, kdy nám počítač vrací hodnotu

$$950\ 706\ 376.(2^{31}-1) = 2\ 041\ 626\ 394\ 607\ 926\ \underline{896}\ ,$$

zatímco ruční výpočet ukazuje, že

$$950\ 706\ 376.(2^{31}-1) = 2\ 041\ 626\ 397\ 607\ 926\ \underline{780}\ .$$

Díky této " zanedbatelné " nepřesnosti nám však počítač po delší simulaci vrací úplně jiné hodnoty než bychom teoreticky očekávali. Zvláště nebezpečná je tato skutečnost z hlediska přenositelnosti generátorů. Poznamenejme však, že se nejedná o nich jiného než o demonstraci omezené přesnosti operace násobení v dolních, nejméně významných bitech. Věc, která je bohužel typická pro většinu mikro a minipočítačů. Pouze malou útěchou nám může být, že z literatury jsou známy situace mnohem horší.

V současné době je běžnou součástí většiny překladačů i řady operačních systémů minimálně generátor pseudonáhodných veličin z R(0,1), často však i generátor z rozdělení jiných. Bohužel, v převážné většině případů chybí nejen popis implementace a výsledky testů jejich chování, nýbrž často dokonce i popis použitého typu generátoru (kongruenční, založený na posuvu registrů apod.) a příslušný funkční tvar. Neprovede-li si uživatel testy takovéhoto generátoru, vystavuje se nebezpečí, že má k disposici některý špatný generátor, jako je tomu v případě překladače IBM BASIC/PC či operačního systému VMS pro počítače DEC VAX, nebo nevhodnou implementaci jinak dobrého algoritmu. Toto nebezpečí zpravidla nevyváží ani snadnost použití generátorů implementovaných v překladačích a operačních systémech, ani jejich rychlost. Základní doporučení před rozsáhlejším používáním takovýchto " neznámých " generátorů je provést si důkladné otestování jejich vlastností.

Podobně téměř ve všech statistických programových systémech je implementován některý generátor náhodných veličin. V tomto případě je sice funkční tvar generátoru zpravidla znám, nicméně většinou i zde chybí výsledky testů dané implementace. Protože uživatel má navíc zpravidla pouze omezené možnosti jak jednoduše v rámci daného systému používat svůj osvědčený generátor, nezbývá mu než se spolehnout na solidnost distributora systému. Na druhé straně je třeba poznamenat, že statistické programové systémy se používají převážně pro vyhodnocování dat a ne pro potřeby rozsáhlejších simulačních studií, takže nehrozí zásadní nebezpečí ani v případě, kdy byl implementován špatný generátor, ani v případě, kdy byl špatně implementován dobrý algoritmus.

V tabulce 1. nalezne čtenář základní informace o generátorech z vybraných statistických programových systémů, operačních systémů, knihoven podprogramů a překladačů. Je však třeba upozornit, že se jedná o informace 1-3 roky staré. Zajímá-li se proto čtenář o generátory implementované ve verzích pro tento rok, nezbývá mu než dopsat jednotlivým distributorům.

Z tabulky 1. vidíme, že převážná většina užitých generátorů je tvaru

$$V_{n+1} = (a V_n) \bmod (2^{31} - 1), \tag{8.1}$$

tj. jedná se o jednoduché multiplikativní kongruenční generátory Lehmerova typu. Testování generátorů tohoto typu se věnovali Fishman a Moore (1982). Na základě výsledků založených na poměrně rozsáhlé baterii testů dospěli k následujícím "optimálním" volbám konstanty a v (8.1), tj.

1) a = 950 706 376;
2) a = 742 938 285;
3) a = 1 226 874 159;
4) a = 62 089 911;
5) a = 1 343 714 438.

Je zajímavé, že z těchto pěti hodnot je pouze první z nich užívána firmou IMSL, zatímco řada jiných distributorů používá takové volby konstanty a, jež se v daném hodnocení umístily mnohem hůře. Na druhé straně je třeba poznamenat, že řada pozdějších autorů argumentovala tím, že požadavky Fishmana a Moora byly příliš přísné a z hlediska praxe "nepříliš podstatné".

Mezi nejčastěji užívané volby konstanty a v tabulce 1. patří a = 16 807. Jedná se o více než 20 let starý návrh Lewise a kol., podrobně testovaný v US Naval Postgraduate School v Monterey, California. Protože ve své době tento generátor, mezi jehož přednosti patří jednoduchost a maximální možná délka cyklu $2^{31}-2$, zdárně prošel všemi tehdy dostupnými testy a osvědčil se i v řadě pratických simulací, některé jeho špatné vlastnosti se

Tabulka 1. Přehled vybraných typů generátorů z překladačů a statistických programových systémů.

Číslo generátoru	Zdroj	Tvar	Poznámka
1	Překladač jazyka APL.	$V_{i+1} = (16807V_i) \bmod (2^{31}-1)$	Tento generátor implicitně užívá i většina programových systémů napsaných v APL, např.Statgraphics.
2	Počítače typu ATARI ST (v OS ROM).	$V_{i+1} = (3141592621V_i+1) \bmod (2^{32})$	-
3	Počítače typu IBM 360/370 , resp.operační systím VMS pro DEC VAX .	$V_{i+1} = ((2^{16} + 3)V_i) \bmod (2^{31})$	Tzv. generátor RANDU. Jedná se o jeden z nejméně kvalitních generátorů jež byl, a leckde bohužel doposud je, široce používán
4	Knihovna podprogramů IMSL, část Stat/Lib, v.1.0 .	a) $V_{i+1}=(950706376V_i) \bmod (2^{31}-1)$	Nejlepší generátor typu $V_{i+1}=(aV_i)\bmod(2^{31}-1)$(dle hodnocení Fishmana a Moora (1982)).
		b) $V_{i+1}=(397204094V_i)\bmod(2^{31}-1)$	Viz LLRANDOMII .
		c) $V_{i+1}=(16807V_i)\bmod(2^{31}-1)$	Viz LLRANDOMI .
5	LLRANDOMI (viz Lewis a další (1988)).	$V_{i+1}=(16807V_i) \bmod (2^{31}-1)$	Jeden z prvních úspěšných generátorů typu $V_{i+1}=(a V_i) \bmod (2^{31}-1)$, navržený již v roce 1969 Lewisem a kol. Postupně byl převzat řadou dalších firem. Po čase mu byly prokázány neoptimální vlastnosti ve vyšších dimenzích.
6	LLRANDOMII viz Lewis a další (1988)).	$V_{i+1}=(397204094V_i)\bmod(2^{31}-1)$	Modifikace generátoru LLRANDOMI navržená v roce 1974 Lewisem a kol. pro odstanění "špatného" chování generátoru LLRANDOMI ve vyšších dimenzích.

Číslo generátoru	Zdroj	Tvar	Poznámka
7	Knihovna podprogramů vyvinutých v JET Propulsion Laboratories v Los Alamos.	$V_{i+1}=(5^{19}V_i) \bmod (2^{48})$	Generátor navržený a široce používaný v laboratořích v Los Alamos na sálových počítačích.
8	Knihovna podprogramů NAG, v.11.	$V_{i+1}=(13^{13}V_i) \bmod (2^{59})$	Značně obskurní volba konstant, jež nebyla nikdy řádně vysvětlena. NAG ani nikdy neposkytl podrobnější informaci o vlastnostech svého generátoru.
9	SASPC v.6.Ø3.	a) $V_{i+1}=(16807V_i) \bmod (2^{31}-1)$	Viz LLRANDOMI.
		b) $V_{i+1}=(397204094V_i) \bmod (2^{31}-1)$	Viz LLRANDOMII.
10	Překladač jazyka SIMULA.	$V_{i+1}=(5^{2p+1}V_i) \bmod (2^n)$	Konstanty p a n jsou voleny dle typu počítače, na němž je jazyk implementován.
11	SPSS/PC v.2, resp. SPSSX v.8.	$V_{i+1}=(16807V_i) \bmod (2^{31}-1)$	Viz LLRANDOMI.
12	Operační systém UNIX.	a) $V_{i+1}=(1103515245V_i + 12345) \bmod (2^{31})$	Tzv. generátor "RAND".
		b) $V_{i+1}=(25214903917V_i + 11) \bmod (2^{48})$	Tzv. generátor "DRAND".

projevily až v polovině 70. let, kdy se lidé začali hlouběji zajímat o mnohorozměrné vlastnosti generátorů. Jedná se o typický příklad situace, kdy požadavky praxe podstatným způsobem změnily pohled na problematiku hodnocení generátorů náhodných veličin. Reakcí autorů generátoru LLRANDOMI byla změna hodnoty a, blíže viz generátor LLRANDOMII. Jak se však ukázalo později, ani tato nová volba nebyla nejoptimálnější. Například v hodnocení Fishmana a Moora se generátor LLRANDOMII umístil až na 10. místě. Na druhé straně "vítězná cesta" generátorů LLRANDOMI a LLRANDOMII přesvědčivě ukazuje, jak velkou roli pro široké nasazení určitého typu generátoru hraje "tradice", dobré jméno autorů a "inzerce", jíž se oběma generátorům při všech možných příležitostech dostalo.

9. GENERÁTOR WICHMANNA A HILLA

Jak jsme se již zmínili dříve, mezi cesty sloužící k vylepšení vlastností generátorů patří jejich kombinování. Typickým generátorem tohoto druhu je i ten, který navrhli Wichmann a Hill (1982). Jelikož se jedná o jeden z nejlepších generátorů pro osobní mikropočítače typu IBM PC jenž je v současné době k dispozici a existují jeho zdařilé implementace ve Fortranu (Wichmann a Hill (1982)) i Pascalu (Wichmann a Hill (1987)), zastavme se u něj podrobněji.

Samotný generátor je definován vztahem

$$(9.1) \qquad U_{i+1} = \left(\frac{X_{i+1}}{30269} + \frac{Y_{i+1}}{30307} + \frac{Z_{i+1}}{30323} \right) \bmod (1),$$

kde X_{i+1}, Y_{i+1} a Z_{i+1} jsou výstupy tří jednoduchých multiplikativních kongruenčních generátorů Lehmerova typu

$$(9.2) \qquad X_{i+1} = (171\ X_i) \bmod (30269),$$

$$(9.3) \qquad Y_{i+1} = (172\ Y_i) \bmod (30307)$$

a

$$(9.4) \qquad Z_{i+1} = (170\ Z_i) \bmod (30323).$$

Generátor (9.1) je založen na tvrzení, že jsou-li X,Y a Z nezávislé náhodné veličiny z $R(0,1)$, potom také veličina $U = (X + Y + Z) \bmod (1)$ je rozdělená jako náhodná veličina z $R(0,1)$.

Mezi přednosti tohoto generátoru patří:

- velmi dlouhá délka cyklu, jež je cca $0{,}7 \cdot 10^{13}$, což je více než 2000 krát více než u generátorů typu (8.1);
- generátor lze implementovat na 16 bitových počítačích v jednoduché aritmetice;
- doposud prováděné testy prokázaly jeho velmi dobré statistické vlastnosti.

Základní nevýhodou je patrně problém rychlosti, jenž je úzce spojen s faktem, že jedna výsledná hodnota se skládá ze tří jiných generátorů.

10. VYBRANÉ ZÁVĚRY TÝKAJÍCÍ SE VÝBĚRU A TESTOVÁNÍ GENERÁTORŮ

Generování pseudonáhodných veličin pomocí algoritmických postupů lze považovat za dostatečně "zralé pole" v tom smyslu slova, že je známo velmi mnoho jak teoretických, tak praktických výsledků o jednotlivých třídách algoritmů. Jak jsme již vícekrát uvedli, v jednotlivých třídách na jedné straně existují velmi dobré generátory, na druhé straně se lze setkat i s generátory s velmi špatnými vlastnostmi. Tato skutečnost vede mimo jiné k následujícím závěrům, na něž by se při výběru a případném testování generátorů nemělo zapomínat.

1. Testování generátorů NENÍ NUTNÉ díky tomu, že existují a jsou více méně běžně k dispozici velmi dobré generátory pseudonáhodných čísel.
2. Testování JE NUTNÉ, neboť v řadě (nejen statistických) programových systémů i překladačů jsou implementovány generátory s velmi špatnými vlastnostmi. Testování by přitom mělo být prováděno pokud možno komplexně a nemělo by se zapomínat na to, že se jedná o úkol pracný a časově značně náročný. Oblast testování generátorů je neustále velmi aktivním polem pro bádání. Testy, teoretické i praktické, neustále procházejí vývojem reagujícím na stále se měnící a nároky zvyšující požadavky uživatelů. Odtud vyplývá skutečnost známá i z jiných oblastí matematické statistiky, totiž, že na dnes běžně užívané a doporučované postupy se může zítra hledět "přes prsty". Viz např. historie generátoru LLRANDOMI. Z tohoto hlediska je proto třeba se dívat na dnes nejlepší generátory jako na postupy, jež s úspěchem prošly rozsáhlými bateriemi testů a prokázaly rozumně dobré vlastnosti v obecných situacích.
3. V každém případě bychom měli DÁT PŘEDNOST používání dobrých generátorů s dokumentovanými vlastnostmi (včetně záporných) před užitím generátorů o nichž nic nevíme. Nicméně i při použití tohoto pravidla je vždy dobré aplikovat alespoň základní empirické a grafické testy, abychom se ujistili o korektnosti naší implementace.
4. Téměř všechny běžně užívané generátory jsou svou povahou celočíselné generátory a pouze nepřímo generátory z $R(0,1)$, přechod mezi nimi je však pouze věcí dělení a užité aritmetiky. Algoritmická povaha většiny běžně používaných generátorů navíc způsobuje, že již a priori každý z nich může být - a zpravidla i je - v určitém smyslu nedokonalý. To znamená, že žádný z generátorů NELZE DOPORUČIT jako ABSOLUTNĚ VHODNÝ pro všechny aplikace.

11. VYBRANÉ PRINCIPY A ZLATÁ PRAVIDLA SIMULAČNÍCH METOD

Veškerý předchozí výklad se zaměřil na jeden ze základních kamenů potřebných pro úspěšné rozvíjení simulačních metod. Zakončeme proto náš příspěvek několika vybranými základními principy a pravidly, jimiž bychom se měli řídit při vlastních simulacích. Je očividné, že jednotlivá pravidla lze pomocí nepodstatných úprav snadno modifikovat též pro problematiku předchozích odstavců.

1. Nikdy nesimuluj, nemusíš-li!

   Jinými slovy to znamená, že nejprve je třeba vždy spočítat (ať již analyticky nebo numericky) vše, co se spočítat dá a využít pro to veškerou informaci o struktuře problému.

2. Pečlivé modelování je velmi důležité!

   Je známým faktem, že nalezení správné odpovědi na špatný problém je k ničemu. Jak

již bylo řečeno, stále častěji se modeluje na počítačích dříve než dochází k reálným experimentům. Nesmí se však přitom nikdy zapomínat na to, že modely jsou především proto, aby se pomocí nich myslelo.

3. Validace modelu a verifikace odpovídajících programů jsou nezbytné!

   Validace modelu lze dosáhnout především pomocí pravidelných konzultací se zákazníky. Verifikace programů pak například jejich aplikací na speciální případy, u nichž jsme schopni zodpovědně posoudit kvalitu výsledků.

4. Nezapomínej, že výsledky získané pomocí simulací jsou, na rozdíl od analytických řešení, pouze odhadem založeným na opakovaných náhodných pokusech!

   Při této příležitosti je absolutně nutné nezapomínat na získání co možná nejlepší představy o přesnosti odhadů.

5. Oblast simulací je velmi široká!

   Každý problém přitom vyžaduje své speciální znalosti, aby bylo možné připravit dostatečně realistický a věrohodný model poskytující rozumné výsledky. Pro splnění tohoto cíle je třeba být připraven a ochoten v každém jednotlivém případě jít do všech zdrojů potřebných informací.

6. Simulační metody jsou interdisciplinární záležitost!

   Nikdy se nesmí zapomínat na to, že simulační metody využívají myšlenek nejen z oblasti teorie pravděpodobnosti a matematické statistiky, nýbrž i numerické matematiky, informatiky, teorie algoritmů atd. Pouze spojením různých interdesciplinárních myšlenek lze dosáhnout skutečně efektivních výsledků.

7. BUĎ SI JIST, ŽE DŘÍVE NEŽ SE SIMULACEMI ZAČNEŠ, VÍŠ NA JAKÉ OTÁZKY CHCEŠ ODPOVÍDAT !!!

LITERATURA

Bratley P. a další (1983). A guide to simulations. Springer, New York.

Brent R.P. (1974). Algorithm 488: A Gausian pseudo-random number generator. CACM, 17, 704-706.

Coveyou R.R. a MacPherson R.D. (1967). Fourier analysis of uniform random number generators. JACM, 14, 100-119.

Fishman G. a Moore L. (1982). A statistical evaluation of multiplicative congruential random number generators with modulus 2^{31}-1. JASA, 77, 129-136.

Gentle J.E. (1981). Portability considerations for random number generators. Ve sborníku konference "13 th Symposium on Interface", editor W.F. Eddy, 158-164. Springer, New York.

Green B.F. a další (1959). Empirical tests of an additive random number generators. JACM, 6, 527-537.

Hurt J. (1982). Simulační metody. SPN Praha.

IMSL (1987). IMSL Stat/Library Users Manual, ver. 1.Ø. IMSL Inc., Houston.

Kennedy W.J. a Gentle J.E. (1980). Statistical computing. Dekker, New York.

Knuth D.E. (1981). The art of computer programming, vol. 2, seminumerical methods. Addison - Wesley, Reading.

Lee Y. K. (1985). Random number generation. BYTE, 10, 426-437.

Lehmer D. (1951). Mathematical methods in large-scale computing units. Annals of Computing Laboratory of Harvard University, 26, 141-146.

Lewis P.A.W. a další (1969). A pseudorandom number generation for the system 360. IBM Systems Journal, 8, 136-146.

Lewis P.A.W. a další (1988). ESSP: Enhanced simulation and statistics package. Wadsworth a Brooks/Cole, Pacific Grove.

Lewis P.A.W. a Orav E.J.(1989). Simulation methodology for statisticians, operations analysts and engineers, Wadsworth a Brooks/Cole, Pacific Grove.

Marsaglia G. (1972). The structure of linear congruential sequences. Ve sborníku konference "Applications of number theory to numerical analysis", editor Zaremba S.K., 249-285. Academic Press, New York.

Marsaglia G. (1985). A current view of random number generators. Ve sborníku konference "16 th Symposium on Interface", editor L.Billard, 3-10. North Holland, Amsterdam.

McLoad A.I.(1985). A remark on algorithm AS 183. Applied Statistics, 34, 198-200.

Ripley B.D. (1983). Computer generation of random variables-a tutorial. International Statistical Review, 51, 301-319.

SAS (1988). Users Guide, ver. 6. SAS Inc., Cary.

Schrage L.(1979). A more portable Fortran based random number generator. Association for Computing Machinery Transactions on Mathematical Software, 5, 132-139.

SPSSX (1986). Statistical Algorithms. SPSS Inc., Chicago.

Walter J. a Lauber J. (1975). Simulační modely ekonomických procesů. SNTL/ALFA, Praha.

Wichmann B.A. a Hill I.D. (1982). Algorithm AS 183. An efficient and portable pseudo-random number generator. Applied Statistics, 31, 188-190.

Wichmann B.A. (1984). Correction to AS 183. Applied Statistics, 33, 123.

Wichmann B.A. a Hill I.D. (1987). Building a random number generator - programming insight. BYTE, 12, 127 - 128.

DALŠÍ DOPORUČENÁ LITERATURA

Brown M. a Solomon H. (1979). On combining pseudorandom number generators. Annals of Statistics, 7, 691-695.

MacLaren M.D. a Marsaglia G. (1965). Uniform random number generators. JACM, 12, 83-89.

Rosenblatt M. (1975). Multiply schemes and shuffling. Mathematics of Computation, 929-934.

Tauseworthe R.C. (1965). Random numbers generated by linear recurrence modulo two. Mathematics of Computing, 19, 201-209.

Toothill J.P.R. (1971). The runs up-and-down performance of Tauseworthe pseudorandom number generators. JACM, 18, 381-399.

Toothill J.R.P. (1973). On asymptotically random Tauseworthe sequences. JACM, 20, 469-481.

Zierler N. (1969). Primitive trinomials whose degree is a Mersenne exponent. Inform. Control, 15, 67-69.