Náhodná čísla a Monte Carlo

Matematické algoritmy (11MAG)

Jan Přikryl

13. přednáška 11MAG
pondělí 6. ledna 2013

verze:2014-01-05 23:33

Obsah

Obsah této přednášky čerpá z klasické přehledové publikace [2] a z možná praktičtěji zaměřené monografie [4]. K dispozici je samozřejmě i celá řada dalších publikací, okrajově je téma zahrnuto i v [1]. V českém jazyce se problematice v rozumném rozsahu věnují snad pouze skripta [7].

1 Stručná historie Monte Carlo metod

Podstatou metody Monte Carlo je simulace hazardních her, jejichž chování a výsledek může být použit ke studiu některých vědecky zajímavých jevů. Zatímco tato podstata není přímo

spojena s počítači, efektivně využit výpočetně simulované hazardní hry jako prostředku seriózní vědecké a technické praxe výrazně je možné až s dostupností moderních výkonných digitálních počítačů. Je zajímavé, a může to někomu připadat až pozoruhodné, že hraní hazardních her nebo náhodný výběr vzorků bude produkovat něco užitečného. Ostatně někteří autoři v době zrodu Monte Carlo metod tvrdili, že Monte Carlo nikdy nebude metodou, použitelnou pro jiné než hrubé odhady číselných veličin.

Patrně první dokumentovaný pokus, jak použít náhodné vzorkování pro nalezení hodnoty integrálu, pochází již z roku 1777.

Příklad 1 (Buffonova jehla). Comte de Buffon tehdy navrhl tento experiment: Mějme jehlu délky ℓ, kterou opakovaně náhodně upustíme na papír s nakreslenou soustavou rovnoběžek, jež se nacházejí ve vzdálenosti $d > \ell$. Jaká bude pravděpodobnost p, že jehla protne některou z rovnoběžek? Comte de Buffon tento experiment provedl a poté také analyzoval matematicky a ukázal, že pravděpodobnost, že jehla protne některou s rovnoběžek, je

$$p = \frac{2\ell}{\pi d}.$$

Po nějaké době navrhl Pierre de Laplace, že by tento experiment bylo možno použít na ověření hodnoty π: Označíme p_N poměr hodů, kdy jehla protla nějakou rovnoběžku, ku všem N hodům. Platí

$$\lim_{N\to\infty} p_N = \frac{2\ell}{\pi D}$$

a tedy

$$\pi = \lim_{N\to\infty} \frac{2\ell}{p_N D}$$

Toto lze opravdu považovat za Monte Carlo metodu na určení hodnoty π. Její rychlost konvergence je ovšem velmi nízká.

Vzorkování se přitom používá už od nepaměti – banky či starověcí výběrčí daní odhadovali počet mincí či jiných komodit tak, že zvážili vybraný vzorek a pak celou hromadu a z poměru usuzovali na celkový počet mincí.

Systematické použití Monte Carlo metod pro řešení reálných technických a vědeckých problémů se prvně objevuje v raných dobách výpočetní techniky a doprovází konstrukci prvního programovatelného počítač na světě (nazýval se MANIAC, z angl. *Mathematical Analyzer, Numerical Integrator and Computer*, a byl umístěn v Los Alamos). Vědci, pracující na projektu první atomové bomby (Stanisław Ulam, John von Neumann, Nicholas Metropolis, Enrico Fermi a další), na tomto počítači řešili stochastickou simulací mimo jiné problém týkající se rozptylu neutronů v různých návrzích struktury atomové bomby a problém odhadu vlastních čísel (kvantové) stacionární Schrödingerovy rovnice.

Dnes již považujeme za samozřejmé, že mnoho problémů v oblasti statistiky či fyziky nebo biologie lze přeformulovat na úlohy statistické inference (usuzování) – jako optimalizační úlohy, jako úlohy pro výpočet mnohorozměrných integrálů nebo jako stochastické simulace. Monte Carlo metody toto umožňují vyčíslit. Zvlášť efektivní jsou v případě vícerozměrných problémů.

Příklad 2 (Optimalizace). Optimalizační problémy jsou problémy typu

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} E(\mathbf{x})$$

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} f(\mathbf{x}|\mathbf{d})$$

kde $E(\mathbf{x})$ označuje očekávanou hodnotu náhodné vektorové proměnné $\mathbf{x}$ a $f(\mathbf{x}|\mathbf{d})$ je podmíněná hustota pravděpodobnosti náhodné vektorové proměnné $\mathbf{x}$ při pozorovaných datech $\mathbf{d}$.

Příklad 3 (Integrace). Dalším příklad využití Monte Carlo metod je výpočet integrálu

$$I = \int_D g(\mathbf{x}) \mathrm{d}\mathbf{x}$$

kde doména integrace D je prostor značně vysoké dimenze (ve finančnictví či fyzice není výjimkou počítat s dimenzemi řádu stovek).

2 Stochastická simulace

Doposud jsme pro řešení matematických problémů používali pouze deterministické numerické metody. Alternativní přístup, jenž se začal rozvíjet společně s nástupem první výpočetní techniky, jsou **stochastické simulace**. Detailní popis tématu jde jako obvykle nad rámec tohoto textu, zmíníme se ale alespoň o základních metodách a o náhodných generátorech, na než tyto metody spoléhají.

Stochastické simulační metody se snaží napodobit nebo ve vybraných případech přímo kopírovat chování systému z reálného světa tak, že využívají náhodnosti jevů – výsledkem simulace jsou vzorky statistické distribuce všech možných výsledků simulace. Vzhledem k náhodnosti, jež je zapojena do celého simulačního procesu, se tyto metody často nazývají Monte Carlo metody.

Stochastické simulační metody jsou užitečné pro zkoumání [1]:

- Nedeterministických procesů
- Deterministických procesů, jejichž popis je tak složitý, že jej nelze sestavit analyticky ■ **tohle ale je numerika, ne? tady heath podle mne nemá pravdu** ■
- Deterministických procesů s vysokou dimenzionalitou (jde například o fyzikální simulace s mnoha stupni volnosti), kde standardní diskretizace, používaná v případě deterministických metod, způsobuje exponenciální nárůst složitosti

Hlavními dvěma požadavky na stochastickou simulaci jsou

- Znalost odpovídajících pravděpodobnostních distribucí
- Dostatečná zásoba náhodných čísel, na jejichž základě při simulaci činíme náhodná rozhodnutí

Znalost relevantních pravděpodobnostních distribucí záleží na naší schopnosti teoreticky popsat nebo empiricky odpozorovat chování simulovaného reálného systému. Jedním z prvních reálných úkolů Monte Carlo metod byla simulace rozptylu neutronů při jejich průchodu stínícím médidem.

Příklad 4 (Průchod neutronu materiálem). K simulaci interakce neutronu s materiálem, jež má fungovat jako stínění, je třeba znát tak zvaný účinný průřez σ, vyjadřující pravděpodobnost, s jakou bude částice z nalétávajícího svazku interagovat s částicí média. Tato hodnota nám v simulaci určí průměrnou délku volného letu neutronu médiem před tím, než dojde ke kolizi s atomy média. Cestu neutronu médiem potom simulujeme na základě posloupnosti náhodných jevů, škálovaných odpovídající pravděpodobností (kolize, pohlcení, odrazu). Simulací dostatečně rozsáhlého počtu trajektorií částic můžeme získat aproximaci výsledné statistické distribuce celkových výsledků, přičemž přesnost této aproximace (bohužel pomalu, ale přece jenom) roste s počtem simulovaných částic.

2.1 Náhodnost

Definovat jednoznačně pojem **náhodnost** je poněkud složité. Fyzikální procesy, o nichž si myslíme, že jsou náhodné (třeba vrh kostkou nebo hod mincí) jsou totiž ve své podstatě deterministické jevy, popsané složitou soustavou pohybových a momentových rovnic pro příslušné počáteční podmínky. V posledních letech se dříve jasná dělící čára mezi deterministickým a náhodným chováním začíná ve světle nových konceptů jako je chaos a chaotické chování dynamických systémů (vzpomeňte si na Lorenzův atraktor) rozmazávat. Složitější nelineární dynamické systémy mohou být totiž extrémně citlivé na své počáteční podmínky a jejich chování v reakci na drobnou změnu počátečních podmínek může být zcela nepředvídatelné i když systém sám o sobě je deterministický – jde o tak zvaný **efekt motýlího křídla** (angl. *butterfly effect*) ■ **vzdálená obdoba špatné podmíněnosti úlohy?** ■.

Příklad 5 (Předpověď počasí). Ačkoliv fyzika dokáže celkem přesně popsat, jak se chová proudění kapalin a tedy i vzdušné proudění naší planety, přesnou předpověď počasí stále nejsme schopni získat pro delší časové období než několik dní. Důvodem je právě citlivost celého systému na drobné změny počátečních podmínek.

2.2 Náhodná čísla

Jedním ze způsobů, jak charakterizovat nepřevídatelnost, kterou spojujeme s náhodností, je říci, že posloupnost čísel je náhodná tehdy a jen tehdy, když pro ni neexistuje žádný kratší popis, než posloupnost sama.

Příklad 6 (Náhodná posloupnost). I když pravděpodobnosti výskytu posloupností $\{1, 2, 3, 4, 5, 6\}$, $\{1, 1, 1, 1, 1, 1\}$, $\{1, 2, 1, 2, 1, 2\}$ či $\{4, 1, 6, 2, 3, 5\}$ mohou být zcela stejné, pouze tu poslední můžeme označit za náhodnou.

V mnoha případech za náhodné považujeme i ne zcela náhodné veličiny, jako je například v teorii hromadné obsluhy čas příchodu požadavku a doba nutná pro jeho zpracování. Sestavit přesný model těchto veličin je totiž příliš složité a navíc tyto veličiny mnohdy ani nelze určit přesně. Studium systémů zahrnujících takové veličiny je potom možno provádět pouze pomocí stochastických simulačních metod.

Druhou významnou vlastností náhodnosti je **neopakovatelnost**. Od náhodné posloupnosti čísel rozhodně neočekáváme, že se bude po nějaké době opakovat: pokud si budeme házet kostkou, očekáváme, že posloupnost hozených hodnot bude zcela nepředvídatelná a že se nezačne v nějakém okamžiku opakovat. Vlastnost neopakovatelnosti u náhodných posloupností je ale u počítačových experimentů v mnoha případech nežádoucí: představte si, že potřebujete ověřit chování nějakého simulačního programu nebo v něm dokonce budete hledat nějaké chyby. V takovém případě potřebuje programátor porušit premisu neopakovatelnosti a stochastický simulační proces spouštět se stále stejnou sekvencí náhodných čísel.

Opakovatelnost je ale dvojsečná zbraň. Statistický významnost stochastikcých simulací závisí na nezávislosti jednotlivých pokusů (let neutronu překážkou v Příkladu 4 by měl pokaždé používat nezávislou posloupnosti náhodných čísel, jinak nebudou výsledky k ničemu). V raných dobách stochastických simulací se pro výběr náhodných čísel používaly knižní tabulky (jako příklad uveďme alespoň [6], až do padesátých let 20. století v podstatě standardní tabulky pro stochastické experimenty)[1] Problém s tabulkami, navíc tak mohutně rozšířenými, byl v tom, že pokud by všichni experimentátoři začali svoji simulaci s náhodnou posloupností začínající na

[1] Doporučuji si pro pobavení přečíst uživatelské recenze na www.amazon.com.

prvním řádku tabulek, všechny stochastické simulace založené na této knize by používaly zcela stejnou náhodnou sekvenci a mohly by vést ke zcela nesmyslným výsledkům.

Jakým způsobem si vybrat vhodný počáteční bod a jak v dnešní době počítače samy generují posloupnosti čísel, jež jsou blízké těm náhodným si povíme v následujícím odstavci.

3 Generátory náhodných čísel

S rozvojem výpočetní techniky převzaly postupně roli generátorů náhodných čísel počítače. Počítačové algoritmy pro generování náhodných čísel jsou ovšem z podstaty výpočetní techniky deterministické, pouze mají tu vlastnosti, že výsledná posloupnost čísel vypadá dostatečně náhodně a neopakovatelně.

Ovšem algoritmus, jenž generuje takovou posloupnost čísel, poskytuje velmi jasný popis (a většinou také velmi krátký, protože chceme, aby generátor náhodných čísel byl co nejrychlejší) toho, jak daná posloupnost vzniká. Z definice náhodnosti je potom jasné, že taková sekvence nemůže být označena jako náhodná, proto se pro označení prvků takto generovaných posloupností vžilo přesnější označení **pseudonáhodná čísla**. Pseudonáhodná posloupnost sice vypadá náhodně, ve své podstatě jde ovšem o opakovatelnou a predikovatelnou sekvenci čísel – pro počítačové odborníky je to příjemná vlastnost, zaručující v případě potřeby možnost ladění chyb v simulačních program a verifikaci výsledků simulace. Důležitým rysem pseudonáhodné poslooupnosti čísel je pak i to, že vzhledem ke konečnému počtu čísel, reprezentovatelných v počítači, se po nějaké periodě musí každá pseudonáhodná sekvence začít opakovat.

Dobrý náhodný generátor by měl mít co nejvíce vlastností z následujícího seznamu [1]:

Generuje náhodný vzorek Generátor by měl splnit statistické testy náhodnosti (například χ^2 test)

Dlouhá perioda Generátor by měl generovat co nejdelší posloupnost čísel před tím, než se hodnoty začnou opakovat (typicky alespoň 2^{32} či 2^{64})

Efektivita Generátor by měl být rychlý a vyžadovat minimum operační paměti (většina simulací potřebuje generovat milióny náhodných čísel)

Opakovatelnost Pro totožné počáteční podmínky generátor produkuje totožnou posloupnost čísel.

Přenositelnost Generátor musí běžet na různých hardwarových platformách a musí přitom poskytovat totožné výsledky.

Splnit všech pět podmínek, uvedených výše, je těžké. Některé běžně používané pseudonáhodné generátory produkují silně korelované posloupnosti, což lze demonstrovat graficky pokud po sobě následující hodnoty použijeme k vykreslení 2D nebo 3D bodového grafu. Takovéto generátory mohou zcela zruinovat význam výsledku stochastické simulace, pokud nejsou použity vhodným způsobem.

3.1 Rané pseudonáhodné generátory

doplnit!

3.2 Kongruenční pseudonáhodné generátory

Již v závěru čtvrté přednášky, když jsme si povídali o významu modulárních výpočtů a kongruencí, jsme si uváděli příklad dnes asi nejrozšířenější formy generátorů pseudonhodných čísel, tedy **lineárního kongruenčního generátoru** (LCG). Takový generátor má tvar

$$x[k+1] \equiv a \cdot x[k] + b \pmod{n},$$

kde a a b jsou nějaká přirozená čísla, počáteční podmínka $x[0]$ se nazývá v překladu náhodné semínko (angl. *seed*) a hodnota n je přibližně (nejčastěji zcela) rovna 2^m, kde m je počet bitů reprezentace přirozeného čísla (m je tedy obvykle 32 nebo 64). Kongruenční generátory generují posloupnosti celých čísel v intervalu $[0, n-1]$ a pro získání hodnot například rovnoměrného rozdělení $\mathcal{U}(0,1)$ je třeba v každém kroku v pohyblivé řádové čárce vydělit $x[k+1]/n$.

Kvalita takového pseudonáhodného generátoru velmi závisí na volbě hodnot a a b a v žádném případě jeho perioda nemůže překročit n. LCG může být poskytovat poměrně kvalitní pseudonáhodné posloupnosti, je ale třeba velmi pečlivě vybírat hodnoty a a b. Většina pseudnáhodných generátorů v počítačových systémech je dnes odvozena od LCG a některé z nich jsou notoricky známé svými nedostatky.

Základním problémem při návrhu hodnot pro lineární kongruenční generátor je dosažení plné periody n. pracoval s plnou periodou n jsou tyto [3]:

1. Čísla a a n jsou nesoudělná.
2. Číslo $a-1$ je dělitelné všemi prvočíselnými faktory čísla n.
3. Číslo $a-1$ je násobek 4, jestliže n je násobek 4.

Kratší perioda LCG není ovšem ještě ten nejhorší problém. Ve většině případů nevhodné volby a a b dojde totiž k tomu, že výsledné hodnoty sekvence padnou do malého počtu hyperrovin v prostoru poměrně nízké dimenze (mnohdy i 3) [5].

Příklad 7 (Multiplikativní kongruernční generátor RANDU). V případě $b = 0$ přejde LCG na multiplikativní generátor Lehmerova typu, používaný ve výpočetní technice už od konce čyřicátých let. Notoricky známým představitelem tohoto typu kongruenčních generátorů je RANDU, definovaný vztahem

$$x[k+1] \equiv 65539 \cdot x[k] \pmod{2^{31}}.$$

Tento generátor je od sedmdesátých let 20. století uváděn jako odstrašující příklad nevhodného návrhu kongruenčního generátoru, neboť generuje velmi deterministickou posloupnost

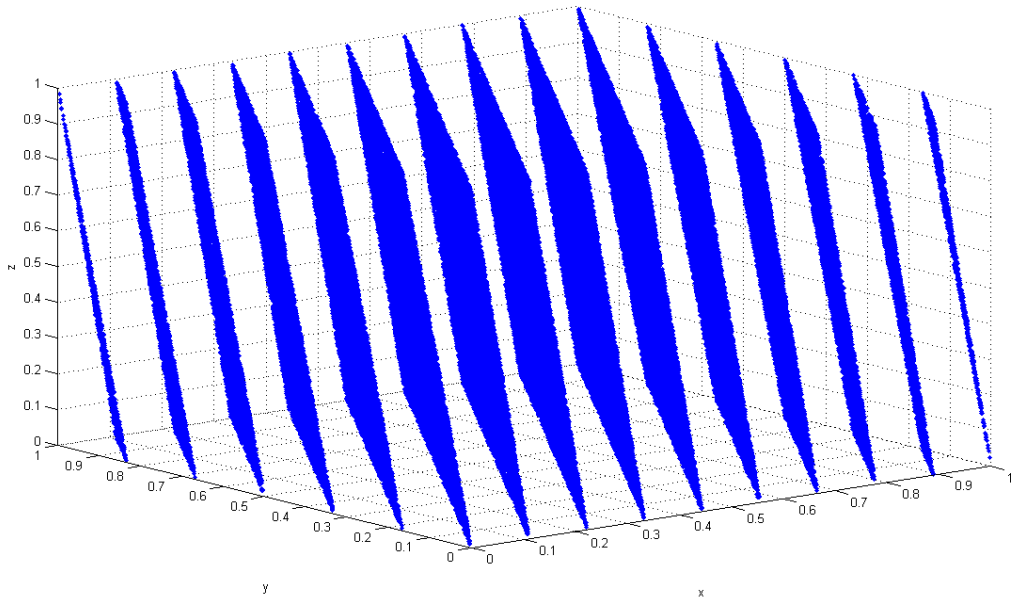
$$x[k+2] \equiv 6\,x[k+1] - 9\,x[k] \pmod{2^{31}},$$

což má za následek, že body ve 3D, tvořené třemi po sobě vzatými hodnotami posloupnosti, leží na celkem 15 rovinách ve třídimenzionálním prostoru. Graf, znázorňující rozmístění jednotlivých bodů, uvádíme na Obrázku 1.

3.3 Fibonacciho generátory

Alternativn9 metodou, produkující rovnou rovnoměrně rozložená čísla v pohyblivé řádové čárce na intervalu 0,1 jsou Fibonacciho generátory, jež generují nové hodnoty jako součet, rozdíl či násobek předchozích hodnot. Typický příkladem je generátor

$$x[k+17] = x[k] - x[k+12].$$



Obrázek 1: 100 002 hodnot vygenerovaných generátorem pseudonáhodných čísel RANDU a použitých po trojicích jako souřadnice bodů zobrazeno jako trojrozměrný graf. Převzato z Wikipedie

Tento generátor má zpoždění 5 a 17. Na výběru zpoždění záleží, jaké bude mít generátor vlastnosti. Hodnota rozdílu může samozřejmě vyjít záporná, v takovém případě přičítáme jedničku, aby se výsledek opět ocitl v požadovaném intervalu 0,1.

Fibonacciho generátor pseudonáhodných čísel vyžaduje více paměti, než LCG. V případě, uvedeném výše, si musíme pamatovat celkem 17 předešlých hodnot. Tyto generátory také vyžadují specifický způsob startování (těch 17 hodnot nelze vybrat zcela náhodně). Na druhou stranu nevyžadují Fibonacciho generátory žádné pomalé dělení v pohyblivé řádové čárce, a dobře navržené Fibonacciho generátory mají velmi dobré statistické vlastnosti. Další výhodou Fibonacciho generátorů je jejich velmi dlouhá perioda, výrazně delší, než perioda kongruenčních generátorů. Důvod je jednoduchý: když u kongruenčního generátoru narazíme na stejný prvek posloupnosti, následné prvky se budou opakovat, v případě Fibonacciho generátoru tomu ale tak není – stejné následné prvky se v posloupnosti objeví až v okamžiku, kdy generátor obdrží totožné všechny zpožděné prvky (v našem případě tedy prvky vzdálené celkem 12 kroků od sebe).

3.4 Nerovnoměrná náhodná čísla

Doposud jsme se zabývali generováním náhodných čísel pocházejících z rovnoměrného rozložení $\mathcal{U}(0,1)$. V případě, že potřebujeme rovnoměrně rozložená pseudonáhodná čísla na jiném intervalu $\mathcal{U}(a,b)$, lze původní hodnoty $x[k]$ jednoduše škálovat vztahem

$$x\prime[k] = (b-a)\,x[k] + a.$$

Výrazně složitější problém na nás ale čeká v případě, kdy bychom rádi vzorkovali z nějakého jiného rozdělení, než $\mathcal{U}(0,1)$. Velmi stručně si nyní ukážeme dva základní postupy, jimiž toho lze dosáhnout.

3.4.1 Metoda inverze

Pokud potřebuje vzorkovat z takového rozdělení pravděpodobnosti, jehož distribuční funkce $F(x)$ je snadno invertovatelná, můžeme použít jednoduchou transformaci hodnot: Nejprve vygenerujeme vzorek z rovnoměrného rozložení na intervalu [0,1] a tento vzorek potom pomocí inverzní distribuční funkce převedeme přímo na vzorek z odpovídajícího pravděpodobnostního rozdělení.

Příklad 8 (Exponenciální rozdělení). Uvažujme exponenciální rozdělení s hustotou pravděpodobnosti

$$f(t) = \lambda e^{-\lambda t}, \quad t > 0$$

a s odpovídající distribuční funkcí

$$F(x) = \int_0^x f(t)\mathrm{d}t = 1 - e^{-\lambda x} = y.$$

Inverzní distribuční funkce je potom

$$F^{-1}(y) = -\frac{\log(1-y)}{\lambda}.$$

Pro generování vzorků z $\mathcal{E}(\lambda)$ potom stačí vygenerovat $y[k] \sim \mathcal{U}(0,1)$ a spočíst $x[k] \sim \mathcal{E}(\lambda)$ jako

$$x[k] = -\frac{\log(1-y[k])}{\lambda}.$$

3.4.2 Zamítací metoda

Mnoho důležitých pravděpodobnostních rozložení má bohužel neinvertovatelnou distribuční funkci – asi nejprominentnějším zástupcem je normální rozdělení pravděpodobnosti, jehož distribuční funkci nelze vyjádřit v uzavřené formě (neznamená to ale, že distribuční funkci $\Phi(x) = 1/2 \cdot \mathrm{erfc}(-x/\sqrt{2})$ a její inverzi nelze spočítat přibližně nebo numericky s nějakou garantovanou chybou – metodu, popsanou v Odstavci 3.4.1 bychom tedy mohli použít, bude to ale velmi pomalé [redacted] **porovnat se Zigguratem** [redacted]).

Jedním ze způsobů (nikoliv ale způsobem nejefektivnějším), jak generovat posloupnost hodnot z libovolného rozdělení $\mathcal{F}$, u něhož jsme schopni analyticky vyjádřit jeho hustotu pravděpodobnosti $f(x)$, ale distribuční funkci $F(x)$ již nikoliv, je tak zvaná **zamítací metoda**, jež pro vygenerování jedné náhodné hodnoty z libovolného rozdělení potřebuje vždy alespoň dvě náhodná čísla. Tato metoda postupuje následovně:

1. Z nějakého rozdělení pravděpodobnosti (může to být rovnoměrné rozdělení, ale nemusí, výhodnější je použít takové rozdělení $\mathcal{G}$, jehož hustota pravděpodobnosti $g(x)$ vhodně aproximuje $f(x)$ až na multiplikativní konstantu M tak, že $\forall x : f(x) \leq Mg(x)$[redacted] **moc koncentrované, rozvést nebo vynechat** [redacted]), navzorkujeme bod x
2. Spočteme hodnotu $f(x)$, neboť hustotu pravděpodobnosti máme dánu analyticky.
3. Vygenerujeme náhodné číslo $u \sim \mathcal{U}(0,1)$
4. Pokud je $u \cdot Mg(x) < f(x)$, akceptujeme x jako vzorek z $\mathcal{F}$. V opačném případě se vracíme na začátek a vzorkujeme nový bod x.

Algoritmus 1 Zamítací metoda vzorkování využívající pouze rovnoměrného rozdělení pravděpodobnosti. Jde pouze o ilustrativní příklad, vhodný pro vzorkování distribucí s „plochou“ hustotou, podobnou rovnoměrnému rozdělení. Vzorkování $\mathcal{N}(0,1)$ není tímto způsobem efektivní

Require: hustota pravděpodobnosti $f(x)$ rozdělení $\mathcal{F}()$
Ensure: $x \sim \mathcal{F}()$
  $f_{\max} \leftarrow \max_x f(x)$
  repeat
    $x \sim \mathcal{U}(-\infty, \infty)$
    $u \sim \mathcal{U}(0,1)$
  until $u \cdot f_{\max} < f(x)$

Algoritmus 1 naznačuje, jak bychom mohli použít zamítací metodu pro vzorkování distribuce s hustotou pravděpodobnosti podobnou hustotě rovnoměrného rozdělení pravděpodobnosti.

Obdobně jako v případě rovnoměrného rozdělení pravděpodobnosti, i v případě normálního rozdělení stačí generovat vzorky ze „základního“ rozdělení $\mathcal{N}(0,1)$. Vzorky $x \sim \mathcal{N}(\mu, \sigma^2)$ lze získat ze vzorků $y \sim \mathcal{N}(0,1)$ transformací

$$x = \sigma y + \mu.$$

4 Monte Carlo integrace

V osmé přednášce jsme se zabývali metodami numerické integrace funkcí jedné proměnné a ukázali si základní kvadraturní vzorce. V metodách výpočetní fyziky či počítačové grafiky se velmi často setkáváme s nutností počítat složité integrály funkcí mnoha proměnných. Pro tyto úlohy není standardní přístup s rovnoměrným rozdělením itegračního intervalu a případným zjemňováním příliš vhodný.

Příklad 9 (Počet vyčíslení vícerozměrné funkce)*.* Budeme-li pro náš kvadraturní vzorec potřebovat vyčíslit v jedné dimenzi integrovanou funkci celkem m-krát, bude v $n \gg 1$ dimenzích tento počet vyčíslení roven m^n. Pokud tedy integrujeme $f(x)$ a vyhodnotíme ji desetkrát, v případě desetirozměrné fyzikální úlohy budeme funkci $f(\mathbf{x})$ vyhodnocovat celkem 10^{10}-krát.

Jediný v praxi použitelný způsob, jak vyhodnocovat vícerozměrné integrály pro počet dimenzí větší, než dva, je **Monte Carlo integrace**. Princip je velmi jednoduchý: Integrovanou funkci $f(\mathbf{x})$ navzorkujeme v celkem m bodech, vybraných náhodně v doméně integrace D a spočteme průměrnou hodnotu $\bar{y}$. Výsledný odhad integrálu je potom roven $I \approx A(D)\bar{y}$, kde $A(D)$ je plocha domény integrace ■ **resp. objem resp. něco jiného** ■. Monte Carlo integrace koverguje velmi pomalu: chyba odhadu klesá s $\mathcal{O}(1/\sqrt{n})$, abychom tedy získali jedno další desetinné místo přesnosti, musíme použít stonásobný počet vzorků. Není proto neobvyklé, že počet vyhodnocení integrandu se pohybuje v řádech miliónů či miliard.

Proč se tedy vlastně Monte Carlo integrace používá, když jde o tak pomalu konvergující postup? (Von Neummann: MC is a method of last resort.) Již jsme si řekli, že pro integraci funkcí v jedné či dvou proměnných postrádá Monte Carlo integrace smysl. Její půvab ale spočívá v tom, že na rozdíl od deterministických postupů není její rychlost konvergence na dimenzionalitě řešeného problému. 1

Příklad 10 (Monte Carlo integrace vícerozměrné funkce)*.* Pro $n = 10^6$ a doménu integrace $D \subset \mathbb{R}^6$ vychází u deterministického přístupu počet vzorků na jednu dimenzi roven 10. Budeme-

li vyžadovat stejnou přesnost, budeme ale při použití deterministického přístupu muset použít vzorků výrazně větší počet ■ **kolik?** ■.

V nejjednodušším případě, kdy $D = [0, 1]$ a $I = \int_0^1 f(x)\mathrm{d}x$, můžeme tento integrál aproximovat jako

$$\bar{I}_m = \frac{1}{m} \left(f(b_1) + \cdots + f(b_m) \right),$$

kde $b_j = j/m$. Jde o aproximaci integrálu, vycházející z původní Riemannovy definice: V každém z m panelů o délce $1/m$ použijeme k odhadu funkční hodnotu na jeho pravém okraji. Budeme-li předpokládat, že funkce f je spojitá a dostatečně hladká, pak se chyba Riemannovy aproximace klesá jako $\mathcal{O}(1/m)$, což je výrazně lepší, než $\mathcal{O}(1/\sqrt{m})$ v případě Monte Carlo integrace (pro dosažení chyby Riemannovy aproximace bychom museli navzorkovat celkem m^2 funkčních hodnot). Použijeme-li pokročilejší deterministická kvadraturní pravidla m-tého řádu, například složené obdélníkové či Simpsonovo pravidlo, bude chyba aproximace integrálu ještě nižší – pro obdélníkové pravidlo bude chyba klesat jako $\mathcal{O}(1/m^2)$, pro Simpsonovo pravidlo dokonce jako $\mathcal{O}(1/m^4)$ (blíže viz [1, odstavec 8.4.1] nebo ■ **desátá?** ■ přednáška MAG).

S rostoucím počtem dimenzí domény D ale deterministická kvadraturní pravidla ztrácí své kouzlo: V případě $D = [0, 1]^{10}$ a $I = \int_{\mathbf{0}}^{\mathbf{1}} f(\mathbf{x})\mathrm{d}\mathbf{x}$ budeme v případě Riemannovy aproximace muset pro dosažení stejného řádu chyby jako výše vyhodnotit celkem $\mathcal{O}(m^{10})$ uzlových vektorů. Pokud použijeme Monte Carlo integraci, navzorkujeme celkem $\mathcal{O}(m^2)$ rovnoměrně distribuovaných vektorů $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{m^2}$ z domény D, tedy celkově $10m$ rovnoměrně distribuovaných hodnot, a chyba aproximace bude stále $\mathcal{O}(1/m)$ bez ohledu na dimenzi problému.

Poznamenejme, že ačkoliv chyba Monte Carlo integrace klesá stejným řádem nezávisle na dimenzi, při rostoucím počtu dimenzí se u této metody musíme potýkat se dvěma problémy, jež vyplývají z podstaty této metody:

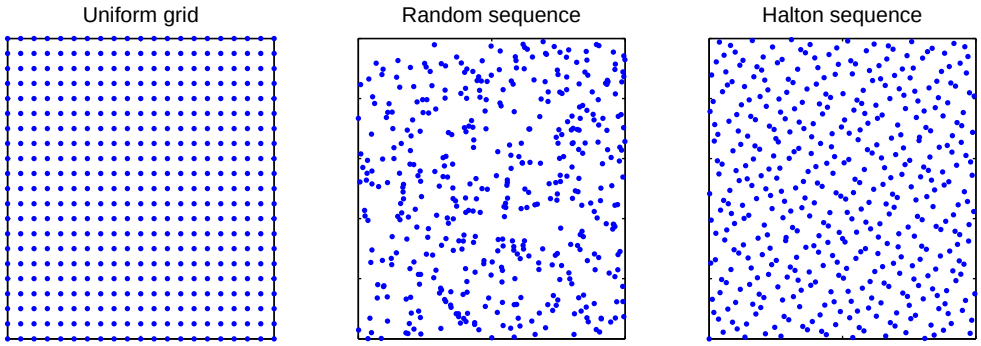
1. Je-li doména D rozlehlá, rozptyl σ^2, tedy míra „uniformity“ integrované funkce f v celém regionu D, může být při rovnoměrném vzorkování velmi vysoký.
2. Vzorkovat rovnoměrně doménu D obecného tvaru nemusí být vždy možné.

Řešením těchto problémů je metoda **vzorkování podle významnosti** (angl. *importance sampling*), kdy vzorky $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ z domény D vzorkujeme z nerovnoměrného rozdělení pravděpodobnosti $\pi(\mathbf{x})$, jež s vyšší pravděpodobností generuje vzorky ve „významných“ částech regionu D. V takovém případě lze integrál I aproximovat jako

$$\hat{I}_m = \frac{1}{m} \sum_{j=1}^{m} \frac{f(\mathbf{x}_j)}{\pi(\mathbf{x}_j)},$$

přičemž rozptyl takového statistického odhadu bude $\sigma_\pi^2 = \mathrm{var}(f(\mathbf{x})/\pi(\mathbf{x}))$. V ideálním případě, kdy je f nezáporná a I konečné, bychom mohli zvolit $\pi(\mathbf{x}) \propto f(\mathbf{x})$. To ovšem znamená, že f známe, a v praxi tato situace není příliš pravděpodobná. Budeme proto spíše hledat takové π, jež je dostatečně podobné f a nasměruje více vozrků do oblastí, kde má f vysokou absolutní hodnotu. Generovat ovšem vzorky z takové spojité distribuce může být docela složité.

Efektivitu Monte Carlo integrace lze zvýšit několika dalšími různými metodami – použitím kvazináhodných posloupností, které rozmísťují vzorkovací body ve doméně integrace rovnoměrněji, než náhodně, ale nikoliv zcela deterministicky, či **vzorkováním po částech** (angl. *stratified sampling*), kdy doménu D dělíme buď předem na několik subdomén a celkový integrál počítáme jako součet dílčích integrálů, nebo doménu dělíme rekurzivně (obdobně jako u adaptivních kvadraturních pravidel) na základě odhadu rozptylu integrované funkce.



Obrázek 2: 441 hodnot vygenerovaných jako rovnoměrná mřížka 21×21 bodů, psudonáhodným generátore a Haltonovou kvazináhodnou sekvencí

5 Kvazináhodná čísla

Přes všechnu snahu, kterou věnujeme vývoji algoritmů, generujících co nejnáhodnější pseudonáhodná čísla, je vhodné si uvědomit, že opravdová náhodnost není vždy nutností. Pro určité vypy aplikací, jako je například Monte Carlo integrace, je podstatně důležitější dosáhnout co nejrovnoměrnějšího náhodného pokrytí vzorkované domény integrace, než to, jestli jsou vzorky opravdu zcela náhodné. Jedním z problémů opravdu náhodných posloupností čísel je totiž to, že jejich hodnoty se mohou v některých oblastech náhodně shluknout a celkové pokrytí vzorkovaného intervalu je tak poměrně nerovnoměrné, viz Obrázek 2. Druhým extrémem je použití zcela rovnoměrné mřížky bodů (jako v případě deterministických kvadraturních algoritmů), jak jsme ale viděli v předchozím odstavci, tento postup nelze efektivně škálovat do vyšších dimenzí.

Kvazináhodné posloupnosti čísel (angl. *quasi-random sequences*) představují jakýsi kompromis mezi zcela rovnoměrným a zcela náhodným pokrytím. Tyto posloupnosti nelze považovat za náhodné, jde o deterministické posloupnosti, jež jsou zkonstruovány tak, aby poskytovaly pro daných k vzorků co nejrovnoměrnější pokrytí vzorkovaného intervalu a zároveň aby vypadaly „dostatečně náhodně“. Jednotlivé prvky kvazináhodných posloupností se sobě záměrně vyhýbají, aby se předešlo shlukům, jež pozorujeme u náhodných posloupností.

Rozdíly všech tří zmíněných přístupů ke vzorkování ukazuje Obrázek 2.

Kvazináhodné posloupnosti, někdy též označované za posloupnosti s nízkou diskrepancí (angl. *low-discrepancy series*) se v dnešní době s oblibou používají v Monte Carlo metodách, vyžadujících dostatečně rovnoměrné pokrytí vzorkované domény a nekladou takový důraz na statistické vlastnosti vzorkovaného souboru – jde například o integraci či náhodné prohledávání. Nejsou naopak vhodné pro statistické simulace, kde velmi záleží na nezávislosti jednotlivých vzorků.

V případě integrace zvýší použití kvazináhodné posloupnosti významně rychlost konvergence, chyba nyní klesá s $\mathcal{O}(1/n)$, bohužel rychlost konvergence se stává závislou na dimenzi řešené úlohy (teoretická horní hranice chyby je pro d dimenzí nepříjemných $\mathcal{O}((\ln n)^d/n)$, v praxi ale v mnoha případech vycházejí i pro $d > 100$ chyby blíže k $\mathcal{O}(1/n)$) a reálná rychlost konvergence proto s rostoucí počtem rozměrů klesá ■ **jak? doplnit** ■.

Reference

[1] HEATH, Michael T. *Scientific Computing: An Introductory Survey*. 2nd Edition. New York: McGraw-Hill, 2002, 563 s.

[2] KALOS, Malvin H. a Paula A. WHITLOCK. *Monte Carlo Methods*. 2. vyd. Weinheim: WILEY-VCH, 2008, 215 s. ISBN 978-352-7407-606.

[3] KNUTH, Donald E. *The art of computer programming*. Volume 2: Seminumerical Algorithms. 3. vyd. Upper Saddle River: Addison-Wesley, 1998, 762 s. ISBN 02-018-9684-2.

[4] LIU, Jun S. *Monte Carlo strategies in scientific computing*. New York: Springer, 2008, 344 s. ISBN 978-038-7763-699.

[5] MARSAGLIA, George. Random numbers fall mainly in the planes. *Proceedings of the National Academy of Sciences*. 1968, roč. 61, č. 1, s. 25-28. DOI: 10.1073/pnas.61.1.25. Dostupné z: http://www.pnas.org/content/61/1/25.short

[6] THE RAND COPRPORATION. *A million random digits with 100,000 normal deviates*. Santa Monica, CA: RAND Corp., 2001, 628 s. ISBN 08-330-3047-7.

[7] VIRIUS, Miroslav. *Metoda Monte Carlo*. Praha: České vysoké učení technické, 2010, 233 s. ISBN 978-800-1045-954.