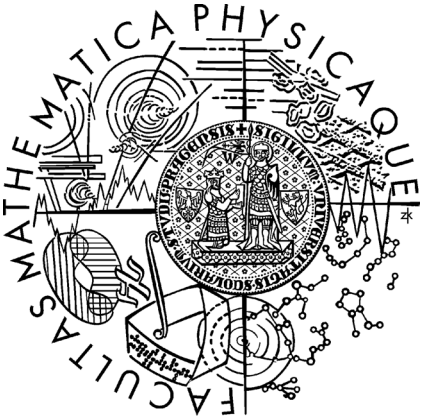
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Adam Kubetta

Simulace ve financích

Katedra pravděpodobnosti a matematické statistiky

Vedoucí bakalářské práce: Doc. RNDr. Jan Hurt, CSc.

Studijní program: Matematika, obecná matematika

2009

Rád bych poděkoval Doc. Hurtovi za vedení bakalářské práce a Dr. Antonínu Slavíkovi za cenné rady ohledně programování.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 7. 7. 2009 Adam Kubetta

Obsah

Název práce: Simulace ve financích
Autor: Adam Kubetta
Katedra: Katedra pravděpodobnosti a matematické statistiky
Vedoucí bakalářské práce: Doc. RNDr. Jan Hurt, CSc.
e-mail vedoucího: hurt@karlin.mff.cuni.cz
Abstrakt: Práce si klade za cíl seznámit čtenáře s metodami Monte Carlo a jejich aplikacemi především ve financích. Převážná část je ale věnována obecnému výkladu, proto může text posloužit také jako učebnice nejen pro matematiky. Důraz je kladen na objasnění souvislostí mezi matematickými modely a aplikovanými algoritmy. Pro správné porozumění se předpokládá aspoň základní znalost pravděpodobnosti a statistiky, ale také programu *Mathematica* společnosti *Wolfram Research*. Stěžejní význam práce totiž spočívá v implementaci metod Monte Carlo v tomto programu. Navíc použití postupů na konkrétních příkladech někdy dokáže problém osvětlit lépe a hlavně rychleji než jakákoliv teorie.
Klíčová slova: Monte, Carlo, finance, simulace, Mathematica

Title: Simulation in Finance
Author: Adam Kubetta
Department: Department of Probability and Mathematical Statistics
Supervisor: Doc. RNDr. Jan Hurt, CSc.
Supervisor's e-mail address: hurt@karlin.mff.cuni.cz
Abstract: The thesis deals with Monte Carlo methods with applications to finance. Most of the thesis is devoted to the general ideas, so the text may serve as a compendium in the topic. The relationship between the models and corresponding algorithms is emphasized. The reader is supposed to be acquainted with basics of probability and statistics and with the Wolfram Research *Mathematica,* the system of doing Mathematics by computer. The core of the thesis is the implementation of the methods in the mentioned system. Some of the examples will explain more than the theory itself.
Keywords: Monte Carlo, finance, simulations, Mathematica

1 Úvod

Cílem práce je usnadnit orientaci mezi základními simulačními postupy používanými ve finančních modelech a upozornit na vestavěné nebo jen jednoduše implementovatelné funkce v systému *Mathematica*. Zdrojové kódy v textu nejsou příliš komentovány ze dvou důvodů. Předně díky výborné nápovědě k programu a také proto, že většinu metod si může každý průměrný uživatel zapsat posvém (a nejspíš i lépe). Ukázky zdrojového kódu jsou použity hlavně pro svou jednoznačnost. Objektový zápis je často průhlednější než matematická notace. Většina postupů byla převzata z práce P. Glassermana [G04], která poskytuje široký přehled metod Monte Carlo využívaných ve financích.

1.1 Pojem simulace

Následující odstavce jsou zařazeny hlavně proto, aby si čtenář vytvořil aspoň přibližnou představu o významu slova simulace v matematice, který není snadné matematicky popsat. Uvedené představy nejsou pro další výklad směrodatné a slouží jen pro ilustraci.

Simulací můžeme (a v tomto textu budeme) nazývat *metodu, která získává empiricky (experimentálně) informace o vlastnostech nějakého systému z matematického modelu tohoto systému*. Schematicky si tedy můžeme představit simulaci jako zobrazení přiřazující matematickému modelu, který reprezentuje nějaký systém (skutečnost), nějaký objekt v tomto modelu ztotožnitelný s informací o vlastnosti skutečného systému.

Co si představit pod pojmem matematický model? Velice zjednodušeně jde o množinu objektů (popisující skutečné objekty), na nichž jsou zavedeny vztahy a přiřazení. Dodejme jen, že vztahy v modelu často reprezentují jen vypozorované vztahy ve skutečném systému, o kterých pouze předpokládáme, že jsou skutečně platné. Pokud se vyvozené důsledky v dostatečné míře shodují s naší zkušeností, přijímáme jejich předpoklady a dále s nimi pracujeme. Takový je ostatně pricip empirického poznávání. Například samotná pravděpodobnost, která je také pouze matematickým modelem, lze dobře připodobnit jevům ve skutečném světě a pomáhá nám činit závěry s přesností, ke které bychom se jinými postupy jen těžko dopracovali. V žádném případě neodpovídá na otázku, jak (jestli) ve skutečném světě vzniká náhoda, spíše nás učí, jak se (přirozeně, v souladu s naší zkušeností) rozhodovat na základě omezených znalostí a jen velmi slabých předpokladů. Uveďme konkrétní příklad jednoduchého modelu.

Uvažme zobrazení z prostoru spojitých funkcí na intervalu do reálných čísel dané následovně

$$T: C([a,b]) \to \mathcal{R},\ T: f \mapsto f(b)$$

Prostor spojitých funkcí na daném časovém intervalu může například reprezentovat vývoj ceny nějaké akcie, a tedy hodnoty funkcí v pravém krajním bodě intervalu potom informaci, která pro nás může být v jistém smyslu atraktivní. Z praxe víme, že ceny jsou nezáporné, proto se v modelu omezíme na nezáporné funkce. Také se můžeme omezit například jen na akcie mající v čase a cenu 1. Takový model s dosud uvedenými vlastnostmi by byl nicméně stále nezajímavý, neboť z něj nelze vyvodit žádné smysluplné obecné závěry pro celý systém.

Přidejme do daného modelu dále pravděpodobnost a předpokládejme rovnocennost všech možných vývojů ceny, tedy "stejně velké množiny" nějakých vývojů se nám jeví jako "stejně pravděpodobné". Nyní již lze sledovat zajímavější vlastnosti jako například pravděpodobnost jevu, že cena akcie v čase *b* bude menší než předem daná libovolná reálná konstanta. Není obtížné si rozmyslet, že v uvedeném modelu by byla pravděpodobnost takového jevu skoro jistě 0, a tudíž cena akcie by rostla před okamžikem *b* skoro jistě (s pravděpodobností 1) nade všechny meze. Ihned tedy vidíme, že volba takového modelu není adekvátní, neboť zdaleka neodpovídá realitě. Model se však výrazně změní přidáním dalších omezujících podmínek, uvažme například omezenost derivací. Každému matematikovi by však mělo být bližší omezit derivace obecněji nějakým pravděpodobnostním rozdělením.

Tím jsme snad nastínili pojem modelu a jeho zásadní důležitost, ale vraťmě se k pojmu simulace. Ve výše popsaném jednoduchém modelu jsme hledali míru pravděpodobnosti nějakého jevu (a tedy odpovídající vlastnost studovaného systému vývoje cen) a velice snadno jsme došli k výsledku, jež nás zajímal. V praxi však model omezují podmínky, které značně znesnadňují explicitní řešení problému, v některých případech dokonce ani není možné problém řešit analyticky. Většinou nám ale také postačí jen přibližné řešení (odhad) a nějaká informace o spolehlivosti tohoto řešení, resp. o vzdálenosti odhadu od přesného řešení. Proto zopakujme a zdůrazněme, že pojem simulace použijeme až tehdy, když k řešení zmíněného problému v matematickém modelu využíváme prostředky pravděpodobnosti a statistiky, především *zákonů velkých čísel* a *centrálních limitních vět*. Jinými slovy: použitá metoda řešení problému není analytická, ale spíše numerická. U simulací typu Monte Carlo spočívá ve správné volbě dat zastupujících "náhodně získaná data" z teoretických rozdělení použitých v modelu. Ve formální definici pojmu simulace jsme toto naznačili výrazem *empirická* metoda (ve statistice tím rozumíme metoda *odvozená z pozorování*). Při simulacích však nahrazujeme pozorovaná měření skutečnosti, která nemůžeme uskutečnit z kauzálních důvodů, správně volenými daty. Volíme-li data jistým způsobem, mohou se jevit jako (metodami statistiky) nerozlišitelné od dat, která bychom získali měřením.

1.2 Základní vlastnosti simulace

Úlohy, které se řeší simulacemi, mívají zpravidla jediné explicitní řešení nebo optimum reprezentované nejčastěji číslem nebo vektorem. Poznatky z teorie pravděpodobnosti nám umožnují jistými transformacemi *náhodného výběru* (tj. vhodně zvolených nezávislých a stejně rozdělených náhodných veličin, jedna taková veličina nám v modelu reprezentuje možné jevy a ohodnocuje je pravděpodobností) najít analyticky odhad (tzv. *bodový odhad*) řešení opět jakožto náhodnou veličinu, tedy pravděpodobnostní rozdělení nad doménou, do které explicitní výsledek patří. Toto obecně není ideální vyjádření výsledku, avšak teorie nám také říká, že s rostoucí velikostí náhodného výběru může aplikace transformací konvergovat k rozdělení degenerovanému do jediného bodu, což už je kvalitativně stejně cenné řešení jako analytické řešení bez použití pravděpodobnosti. Prakticky nám však stačí výsledek, který tvrdí, že s velmi vysokou pravděpodobností se explicitní řešení nachází ve velmi malé oblasti (tím máme namysli zejména *intervaly spolehlivosti*). Teorie nám také může prozradit, jak velké náhodné výběry volit, abychom dosáhli požadované přesnosti.

Simulační metoda používá stejné transformace jako analogická teoretická metoda založená na pravděpodobnosti, avšak jejím vstupem namísto náhodných veličin s nějakým rozdělením jsou data, která by měla pocházet z tohoto rozdělení, a tedy obdržený výsledek je číslo (vektor) pocházející z teoretického výsledného rozdělení. To většinou neumíme snadno zjistit. Kdyby ano, nemuseli bychom

používat simulaci. V těchto případech ke spočtenému výsledku stačí zjistit, nebo aspoň odhadnout pro lepší představu o jeho spolehlivosti, některé charakteristiky teoretického rozdělení, ze kterého pochází. Známe-li například *rodinu rozdělení*, ze které teoretické rozdělení pochází, můžeme parametry rozdělení odhadovat jejich výběrovými protějšky.

1.3 Srovnávání simulačních metod

Dříve než si ukážeme vlastní metody, musíme si ujasnit, dle jakých kritérií je budeme srovnávat. Z matematického hlediska je nejdůležitějším ukazatelem přesnost výpočtu. Metoda pro nás bude samozřejmě tím lepší, čím méně bude rozptýlené teoretické rozdělení výsledku. Toto lze zřejmě efektivně měřit *teoretickým rozptylem* odhadu výsledku pro daný rozsah náhodného výběru n (u simulace budeme říkat *rozsah dat*). Bez obav z nedorozumění budeme dále používat výraz *rozptyl metody* místo teoretický rozptyl odhadu výsledku získaného danou metodou a podobně různými vlastnostmi metody budeme rozumět vlastnosti odhadu výsledku při použití dané metody. Jak velké n ale volit? Samozřejmě řádově srovnatelné s rozsahem dat, který hodláme prakticky zpracovávat. Chceme-li být teoreticky důslednější, zohledníme dále následující.

Uvažme například pouze *konzistentní* metody, to jest takové, že s rostoucím n konverguje rozptyl odhadu k 0. Potom může teoreticky nastat situace, že uvedené kritérium nerozliší dvě metody, protože pro různá n obdržíme opačné rozhodnutí. V takovém případě by pro nás měla být podstatná *rychlost konvergence* rozptylu. Potom totiž jistě bude existovat určitá mez n_0, od které bude rychleji konvergující metoda stále lepší dle prvního kritéria a s rostoucím n také stále výrazněji, tzn. v relativním poměru rozptylů porovnávaných metod.

Z praktického hlediska pro nás bude dále neméně významná *časová náročnost* metody. Tu lze odhadnout náročností aritmetiky při provádění transformací charakterizujících danou metodu. V dnešní době však výpočetní prostředky využívají jen těžko průhledné optimalizace, navíc si ukážeme metody využívající sofistikovanější tvorbu dat, která se jistě také musí projevit na časové složitosti výpočtu, takže časovou náročnost je přirozeně lepší měřit přímo. K tomu nám dobře poslouží vestavěná funkce *Timing* programu *Mathematica*. Poměr časů potřebných k výpočtu u dvou srovnávaných metod (opět pro pevně dané n) se také nazývá *pracovní poměr* a může samozřejmě záviset na softwaru, v němž algoritmy implementujeme , a tedy i hardwarových prostředcích, které program využívá.

Kombinací obou předchozích kritérií dostaneme takzvanou relativní eficienci danou vztahem

$$E(m_1, m_2) = \frac{t_1 \sigma_1^2}{t_2 \sigma_2^2},$$

kde t_i je čas potřebný k získání výsledku metodou m_i a σ_i^2 je teoretický rozptyl (případně střední čtvercová odchylka) metody m_i, pro $i = 1, 2$.

Další velice důležitou vlastností metody je její *nestranost* (ještě naposledy připomenu, že jde o vlastnost odhadu výsledku získaného metodou). Představme si, že aplikovaný postup nám s rostoucím rozsahem dat n konverguje k explicitnímu výsledku, tedy i střední hodnoty odhadů konvergují ke stejnému výsledku, avšak velice pomalu, bez ohledu na to, jak rychle konverguje rozptyl (pro lepší srovnání směrodatná odchylka). Může se tedy stát, že pro jistá n již dosáhneme nějakého výsledku, který pochází z rozdělení s požadovaným velice malým rozptylem, přesto může být toto řešení značně vychýlené. U nestranných (jinak též nevychýlených) metod přidáváme požadavek

na nestrannost odhadu řešení, tzn. střední hodnota odhadu musí být vždy rovna explicitnímu řešení nezávisle na velikosti vstupních dat n. V takových případech uvedený problém s vychýlením výsledku nenastává. Zřídka se také používají metody, které nejsou nestranné. U takových metod potom požadujeme aspoň velice rychlou konvergenci teoretických středních hodnot odhadu.

1.4 Úmluvy a značení

Ještě než začneme pracovat s matematickými pojmy, dohodněme se na následujícím:

• V celé práci si vystačíme s měřitelnými prostory $\left(\mathcal{R}^k, \mathcal{B}\left(\mathcal{R}^k\right), \lambda\right)$. Jinými slovy budeme pracovat jen s reálnými vektorovými prostory, jejich Borelovskými σ-algebrami a Lebesgueovou mírou na nich, kterou též budeme jednoduše nazývat objem. Objeví-li se kdekoliv mezi předpoklady nějaká množina A (resp. funkce f), automaticky předpokládáme, že A (resp. f) je λ-měřitelná. Prostor funkcí, které mají konečný Lebesgueův integrál v mocnině p na množině M (prostor integrovatelných funkcí) budeme značit $\mathcal{L}_p(M)$.

• Pro indikátor jevu (např. $x > c$), neboli funkci, která nabývá hodnoty 1 pokud jev nastává (argument je pravdivý) a jinak nabývá hodnoty 0, budeme používat symbol $\mathbb{1}_{(x>c)}$. V *Mathematice* se tato funkce jmenuje *Boole*. V pravděpodobnosti obvykle místo konrétního x pracujeme s náhodnou veličinou a indikátor se pak stává také náhodnou veličinou s alternativním rozdělením.

• Pro charakteristickou funkci množiny A (někdy říkáme také indikátor množiny), tj. funkci, která nabývá hodnoty 1 na A a jinde nabývá hodnoty 0, budeme používat symbol χ_A. Samozřejmě lze definovat jako $\chi_A(x) = \mathbb{1}_{(x \in A)}$.

• Náhodné veličiny s rovnoměrným rozdělením na nějaké množině M označíme zpravidla $\mathcal{U}(M)$ (uniformní distribuce).

• Stříškou nad výrazem budeme naznačovat, že jde o odhad hodnoty tohoto výrazu ($\hat{P}$ je odhadem P). Mějme stále napaměti, že odhad je náhodná veličina a především funkce nějakého náhodného výběru $\mathbb{X}_n = \{X_1, \ldots, X_n\}$. Často nás bude zajímat hlavně závislost odhadů na rozsahu výběru n a limitní chování pro $n \to \infty$. Správně bychom tedy měli psát $\hat{P}_{\{X_1, \ldots, X_n\}}$, ale používat budeme obvykle značení $\hat{P}_n$, nebo jen $\hat{P}$.

2 Generování náhodných dat

Jádrem každé simulace typu Monte Carlo je vytváření dat, která co nejlépe zastupují realizaci náhodného výběru. Stručně připomeňme, že *náhodným výběrem* $\mathbb{X}_n$ se rozumí soubor n nezávislých náhodných veličin (obecněji náhodných vektorů) $\{X_1, \dots, X_n\}$ se stejným rozdělením ρ, a mluvíme-li o *realizaci náhodného výběru*, máme na mysli konkrétní data, která obdržíme vybíráním prvků z daného rozdělení ρ, tedy nějakou n-tici nezávisle náhodně vybraných čísel (resp. vektorů) $\mathbb{x}_n = \{x_1, \dots, x_n\}$.

Pro lepší představu podotkněme, že se vlastně jedná o postup v jistém smyslu opačný ke standardnímu použití statistických metod. Obvykle totiž máme dán soubor dat (pozorování, měření) a pomocí statistiky testujeme, jaké rozdělení těmto datům nejlépe odpovídá, jinými slovy z jakého rozdělení byla náhodně vybrána, neboli o jakou realizaci náhodného výběru jde. U metod Monte Carlo naopak znalost rozdělení předpokládáme a uměle vytváříme data, po kterých požadujeme, aby se co nejvíce podobala pozorováním pocházejícím z daného rozdělení. Podobou zde rozumíme především nezávislost a respektování daného rozdělení. Ve statistice existuje řada testů nezávislosti dat. Základní postupy lze najít např. v [H82]. Respektování daného rozdělení zhruba znamená, že pro rozsáhlá data z nějaké nosné množiny a zvolenou podmnožinu odpovídá přibližně četnost dat ze zvolené podmnožiny pravděpodobnostní míře této podmnožiny. Existují i míry (nazýváme je *diskrepance*) měřící tuto vlastnost. Na jejich základě jsou vybudovány tzv. metody *quasi Monte Carlo*, o kterých v krátkosti pohovoříme v samostatné kapitole.

Procesu vytváření dat s popsanými vlastnostmi budeme říkat *generování náhodných dat* a nástroji pro jejich vytváření *generátor*. Ačkoliv se může na první pohled zdát, že musí jít o proces, který vytváří data zcela neočekávaně, ve skutečnosti prakticky využíváme jen procesy zcela deterministické, proto někdy budeme přesněji mluvit o generátorech *pseudonáhodných dat*. Protože při řešení úloh metodami Monte Carlo využíváme statistiku, stačí nám, aby data vypadala náhodně z pohledu statistiky, což samozřejmě není snadný úkol a existuje rozsáhlá teorie zabývající se touto kamufláží.

Pro začátek si uvědomme, že vhodnou transformací náhodného výběru z rovnoměrného rozdělení na intervalu [0,1] lze dostat náhodný výběr z libovolného rozdělení. Samozřejmě bychom mohli stejně dobře volit jiný interval a jiné rozdělení, ale náhodná veličina s ronoměrným rozdělením $\mathcal{U}([0,1])$ se asi nejsnáze transformuje a příslušný generátor většinou patří do základní výbavy mnohých programů.

Dodejme ještě, že prostory, ze kterých data vybíráme, nemusí mít nutně konečnou (ani spočetnou) dimenzi. Ve složitějších úlohách například pracujeme s náhodnými procesy se spojitým časem. Tématem generování dat se budeme zabývat jen okrajově. Zaměříme se spíš na to, aby se čtenář seznámil se způsoby generování různých typů dat v *Mathematice*. Za tímto účelem lze také doporučit pročtení nápovědy k programu *Mathematica*, která již sama o sobě je v rámci možností výbornou učebnicí nejrůznějších matematických postupů, včetně generování pseudonáhodných dat. Konkrétně čtenáře odkážeme na průvodce [W08R] dostupného samostatně také na webu.

2.1 Vlastnosti náhodných dat a generátorů

Jak jsme již uvedli, prvořadým cílem při generování náhodných dat bývá zpravidla možnost získávat náhodná data z rovnoměrného rozdělení $\mathcal{U}([0,1])$. Mezi základní postupy řadíme *lineární kongurenční generátory*, které generují rekurentně čísla $u_i \in [0,1]$ podle předpisu

$$x_{i+1} = (a\, x_i + c) \bmod m,$$
$$u_{i+1} = x_{i+1} / m,$$

kde $a, c, m \in \mathcal{N}$ a na počátku volíme tzv. *semínko* x_0. Na kongurenčních generátorech si ukážeme řadu vlastností, které hrají v simulačních úlohách důležitou roli.

Předně si uvědomme, že uvedený generátor dokáže vytvářet nejvýše m různých hodnot. To nemusíme považovat za velkou nevýhodu, pokud pracujeme s dostatečně velkým rozlišením, tedy například s m řádově aspoň v milionech. Vždyť prakticky čísla rozlišujeme jen do jistého řádu. Jistě ale každého ihned napadne, že již lineární transformací, kterou bychom chtěli získat náhodnou veličinu z mnohem širšího intervalu, se stane rozlišení úměrně hrubějším.

Každý kongurenční generátor zřejmě v nějakém kroku musí vrátit zbytek x_{i+1}, který se vyskytl v rekurentní posloupnosti dříve. Počet kroků k nejbližšímu stejnému zbytku určuje *periodu kongurenčního generátoru*. Generátoru, který během $m-1$ kroků nezopakuje žádný zbytek, říkáme *generátor s plnou periodou*. Samozřejmě vyžadujeme, aby měl generátor co možná největší délku periody.

Nejvíce ale vždy dbáme na *náhodnost* dat, tedy podobnost s pozorováními, o které jsme mluvili v úvodu kapitoly. U kongurenčních generátorů závisí na volbě parametrů a, c, m. V zásadě se při výběru parametrů řídíme teoretickými vlastnostmi a statistickými testy. Protože se kongurenčí generátory v praxi hodně využívají, existuje rozsáhlá teorie, která se jim věnuje. Uvedli jsme jen základní model a hlouběji se teorii věnovat nebudeme.

Důležitou vlastností je *reprodukovatelnost* dat. Volbou stejné inicializační hodnoty (semínka) obdržíme stejnou sekvenci dat. Přestože v této práci při implementaci algoritmů nebudeme příliš brát ohled na paměťové nároky a s generovanými daty budeme pracovat v celku, může být obecně možnost reprodukovat data důležitá při srovnávání metod, pokud bychom pracovali s velkými rozsahy dat, která by bylo nutné zpracovávat sekvenčně. My ale budeme celé realizace náhodných výběrů ukládat v paměti a ponecháme si tak možnost aplikovat data opakovaně v libovolné proceduře.

S reprodukovatelností ještě souvisí *přenositelnost*. Generátor by měl pracovat stejným způsobem na libovolné platformě, nezávisle na reprezentaci čísel. Některé generátory totiž za účelem urychlení využívají specifických vlastností reprezentace čísel na daném stroji.

Tím se dostáváme k největší přednosti kongurenčních generátorů, totiž *rychlosti* generování. Stěží najdeme rychlejší možnost generování dat se srovnatelnými vlastnostmi. Pro dosažení praktických výsledků potřebujeme generovat data nejméně v řádech tisíců. Na počítači, v nemž byl psán tento text (s dvoujádrovým procesorem *Core2 T5500* pracujícím na frekvenci 1.66 GHz), trvá programu *Mathematica* generovat 10 milionů čísel z rovnoměrného rozdělení $\mathcal{U}([0,1])$ přibližně půl vteřiny. Toto tvrzení jako jedno z mála doložím i důkazem.

```
Table[Timing[RandomReal[{0, 1}, 10^7];], {5}] // TableForm
```

```
0.5    Null
0.469  Null
0.469  Null
0.5    Null
0.516  Null
```

■ 2.2 Generování z rovnoměrného rozdělení

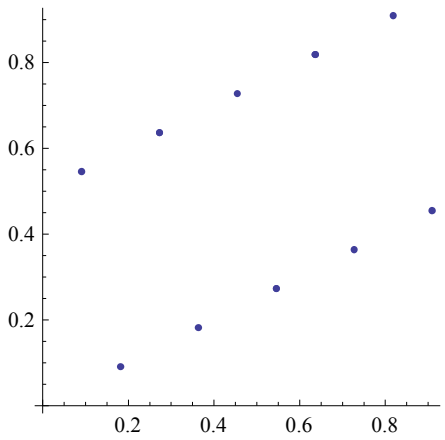
Než přejdeme k praktičtějšímu výkladu, uveďme zásadní nevýhodu jednoduchého lineárního kongurenčního generátoru, o kterém byla řeč. Při simulacích často potřebujeme generovat mnohorozměrná data (představujme si nyní prvky $[0,1]^d$). Za předpokladu generování skutečně náhodných dat $u = \{u_1, u_2, u_3, \ldots, u_n\}$ z intervalu [0,1] bychom mohli za náhodná data například z $[0,1]^2$ považovat vektory $v = \{\{u_1, u_2\}, \{u_2, u_3\}, \ldots, \{u_{n-1}, u_n\}\}$. Pokud ale sekvenci u generujeme uvedeným lineárním kongurenčním generátorem, prvky v musí ležet na několika rovnoběžných ekvidistantně vzdálených přímkách. Tento fenomén je běžně známý jako *mřížková struktura*. V praxi se proto používají raději *kombinované generátory* vznikající sčítáním více kongurenčních generátorů, *násobné rekurzivní* generátory, které dostaneme použitím rekurzí vyšších řádů, *nelineární* postupy a jiné složitější metody.

Jak se můžeme přesvědčit, program *Mathematica* standardně využívá sofistikovanějších metod, a jestliže tudíž chceme vidět mřížkovou strukturu, musíme jej přinutit používat zmíněné jednoduché metody generování. Pro vysvětlení, parametry *"Multiplier"*, *"Increment"* a *"Modulus"* odpovídají pořadě parametrům a, c, m v modelu, který jsme zavedli v předcházející podkapitole. Funkce *SeedRandom* určuje obecně hodnotu semínka při generování, pokud ji nenastavíme, mění se při každém volání generátoru.

```
U = BlockRandom[SeedRandom[1,
     Method → {"Congruential",
       "Multiplier" → 6, "Increment" → 0, "Modulus" → 11
      }]; RandomReal[1, 14]];
```

```
{{Partition[U, 2, 1],
   ListPlot[Partition[U, 2, 1], AspectRatio → 1]
  }} // TableForm
```

```
0.545455   0.272727
0.272727   0.636364
0.636364   0.818182
0.818182   0.909091
0.909091   0.454545
0.454545   0.727273
0.727273   0.363636
0.363636   0.181818
0.181818   0.0909091
0.0909091  0.545455
0.545455   0.272727
0.272727   0.636364
0.636364   0.818182
```



Neodbočujme ale od tématu a začněme jednoduššími příklady. Pokud budeme chtít generovat (kvalitně) n náhodných vektorů z rovnoměrného rozdělení na $[a,b]^d$, $a, b \in \mathcal{R}$, $d \in \mathcal{N}$ je dimenze reálného vektorového prostoru, pak jednoduše voláme funkci *RandomReal* s následujícími parametry.

```
n = 5;
d = 3;
{a, b} = {0, 1};
v = RandomReal[{a, b}, {n, d}];

v // TableForm
```

```
0.880635  0.715954 0.458529
0.820312  0.12291  0.664543
0.69768   0.881498 0.165358
0.0815178 0.44779  0.205737
0.866485  0.596433 0.990833
```

Do proměnné v jsme uložili seznam n náhodných vektorů (matici typu $n \times d$) a hodnoty vypsali ve formě tabulky.

2.3 Inverzní transformace

Mathematica sice obsahuje ve standardní výbavě funkce generující data přímo z široké palety nejrůznějších distribucí (spojitých i diskrétních), o nich ještě bude řeč později, ale někdy nám ani tyto nemusí stačit. Vraťme se proto opět na chvíli k teorii. Předpokládejme, že dokážeme generovat data z rovnoměrného rozdělení $\mathcal{U}([0,1])$ a chtěli bychom generovat data z rozdělení s distribuční funkcí F. Pokud je F rostoucí, pak náhodná veličina

$$X = F^{-1}(U),\ U \sim \mathcal{U}([0,1]),$$

kde F^{-1} je inverzní funkce k F, má právě ono požadované rozdělení, neboť z vlastností distribuční funkce platí pro každé $x \in \mathcal{R}$

$$\begin{aligned} \mathrm{P}(X \le x) &= \mathrm{P}(F^{-1}(U) \le x) \\ &= \mathrm{P}(U \le F(x)) \\ &= F(x)\ . \end{aligned}$$

Pro obecnou distribuční funkci F nahrazujeme inverzi kvantilovou funkcí

$$F_{-1}(u) = \inf\{x, F(x) \ge u\},\ u \in (0,1)\ ,$$

která se pro rostoucí funkce F shoduje s inverzní funkcí a navíc je dobře definovaná i pro skokové a neklesající distribuční funkce. Pokud F určuje rozdělení, které má spojitou hustotu f, potom je F zřejmě rostoucí v bodech, kde $f > 0$.

Na základě popsaných znalostí můžeme v *Mathematice* definovat pomocné funkce, které umožní generovat data z rozdělení s hustotou ρ na zvoleném intervalu. Zavedeme nejprve funkci *TransformToDensity*, která se pokusí najít inverzní transformaci tak, že zadanou hustotu ρ na zadaném intervalu $[a,b]$ analyticky integruje na distribuční funkci a následně invertuje, poté vybere první možné řešení zobrazující číslo 1 na b, a dále zavedeme funkci *RandomRealFromDensity* generující data za pomocí inverzní transformace získané prvně zmíněnou funkcí.

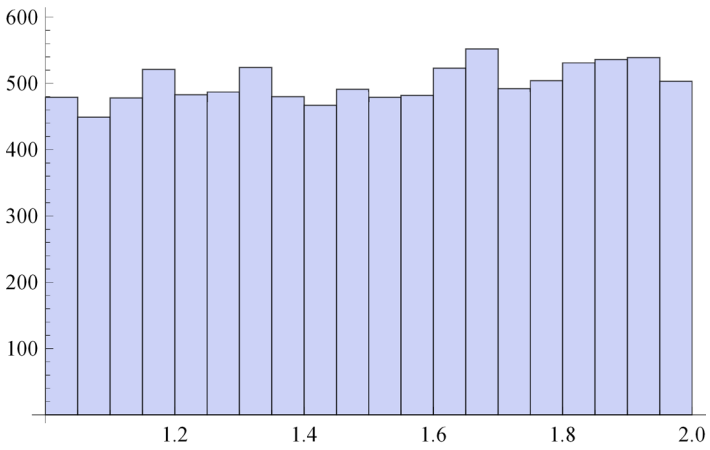
```
TransformToDensity[ρ_, {a_, b_}, x_] :=
  Module[{u, z},
   Select[
      z /. Solve[∫_a^z ρ[t] dt == u ∫_a^b ρ[t] dt, z],
      (# /. u → 1) == b &
    ][[1]] /. u → x];

TransformToDensity[ρ_, {a_, b_}] :=
  TransformToDensity[ρ, {a, b}, #] &;

RandomRealFromDensity[ρ_, {a_, b_}, n___] :=
 RandomReal[{0, 1}, n] // TransformToDensity[ρ, {a, b}]
```

Pokud integrál přes $[a,b]$ z ρ není roven 1, tj. pokud není hustota zadána korektně, vypořádá se s tím funkce tak, že nejprve hustotu normuje. Zdůrazněme, že deklarované funkce zafungují správně jen na ne-příliš složité vstupní funce udávající hustotu, zkrátka na vstupy, které *Mathematica* analyticky upočítá. Samozřejmě bychom mohli funkce definovat daleko důmyslněji, například aby dopočítávali nějaké aproximace inverze, nebo aspoň ošetřit potenciálně nepříjemný výstup (ve smyslu neočekávaný a velice rozsáhlý, takže může *Mathematica* věnovat zbytečně spoustu času formátování grafiky výstupu). Jedná se nám ale spíš jen o jednoduchou pomůcku, kterou použijeme na rozsáhlá data až v případě, kdy budeme mít jistotu, že se inverze spočítá. Uveďme nějaké příklady použití.

```
TransformToDensity[2 &, {0, 1}][x]
TransformToDensity[(2 #) &, {0, 1}][{0, 0.04, 1, x}]
Histogram[RandomRealFromDensity[1 &, {1, 2}, 10000]]
```

```
x
```

$$\{0,\ 0.2,\ 1,\ \sqrt{x}\}$$



V prvním příkladu jsme demonstrovali, že ρ musí být zadána v podobě ryzí funkce (*pure function*) a že funkce *TransformToDensity* vrací stejné výsledky pro $c\rho$, kde $c \neq 0$ je konstanta. V druhém příkladu jsme ukázali, že se funkce *TransformToDensity* automaticky mapuje na seznamy (má jako funkce jedné proměnné implicitně atribut *Listable*) a pracuje správně i s číselnými hodnotami (to není úplně samozřejmé, *Mathematica* může např. nejprve dosadit za proměnnou a poté se pokoušet derivovat číslo). Posledním příkladem jsme histogramem ověřili, že funkce *RandomRealFromDensity* skutečně nagenerovala data z požadovaného rozdělení s konstantní hustotou na intervalu [1,2].

2.4 Zamítací metoda

Asi častěji než inverzí transormace se využívá metoda zamítání. Předpokládejme, že dovedeme generovat data z rozdělení s hustotou ρ na nějaké obecné množině $\mathcal{M}$ a chtěli bychom dostat data z rozdělení s hustotou f na $\mathcal{M}$. Jestliže funkce splňují pro nějakou konstantu c

$$f(x) \leq c\,\rho(x) \quad \forall\, x \in \mathcal{M},$$

pak generujeme prvky z rozdělení náhodné veličiny X s hustotou ρ, ale přijímáme je s pravěpodobností $p(X) = f(X) / c\,\rho(X)$. Funkce p vrací pro prvky $\mathcal{M}$ díky podmínce pro f a ρ skutečně hodnoty z intervalu [0,1] a kritérium přijetí (zamítnutí) lze proto implementovat tak, že k prvku z rozdělení veličiny X generujeme číslo z rozdělení náhodné veličiny $U \sim \mathcal{U}([0,1])$ a zamítáme pokud $U > p(X)$. Pokud je generovaný kandidát z rozdělení X zamítnut, vybírají se opakovaně další prvky z rozdělení X i U, dokud nedojde k přijetí.

Nyní ukážeme, že přijaté prvky opravdu tvoří data z rozdělení s hustotou f. Využijeme přitom poznatků o počítání s podmíněnou střední hodnotou. Označme Y náhodnou veličinu, kterou modelově vrací popsaný algoritmus, pak pro libovolnou $\mathcal{A} \subset \mathcal{M}$ máme

$$P(Y \in \mathcal{A}) = P\left(X \in \mathcal{A} \;\middle|\; U \leq \frac{f(X)}{c\,\rho(X)}\right) = \frac{P\left(X \in \mathcal{A},\, U \leq \frac{f(X)}{c\,\rho(X)}\right)}{P\left(U \leq \frac{f(X)}{c\,\rho(X)}\right)}.$$

Nyní si uvědomme, že pro pevnou hodnotu $x \in Dom(X)$ je z vlastností $\mathcal{U}([0,1])$

$$P\left(U \leq \frac{f(x)}{c\,\rho(x)}\right) = \frac{f(x)}{c\,\rho(x)}.$$

Proto s ohledem k hustotě rozdělení náhodné veličiny X integrujeme ve jmenovateli přes celý nosič veličiny X a v čitateli jsme omezeni podmínkou $X \in \mathcal{A}$. Celkově dostáváme

$$P(Y \in \mathcal{A}) = \frac{\int_{\mathcal{A}} \frac{f(x)}{c\,\rho(x)}\, g(x)\, dx}{\int_{\mathcal{M}} \frac{f(x)}{c\,\rho(x)}\, g(x)\, dx} = \frac{\int_{\mathcal{A}} \frac{f(x)}{c}\, dx}{\frac{1}{c}} = \int_{\mathcal{A}} f(x)\, dx.$$

Zároveň si můžeme všimnout, že výraz ve jmenovateli udává pravděpodobnost přijetí kandidáta. Protože f i ρ jsou hustoty pravděpodobnostního rozdělení, hodnota $1 / c$ musí ležet v intervalu [0,1]. Při aplikaci metody proto hledáme nejen hustotu ρ, ze které dovedeme generovat dostatečně rychle, ale i nejmenší přípustnou konstantu c (nejblížší hodnotě 1).

2.5 Generování z normálního rozdělení

Stručně připomeneme, že normální rozdělení s parametry μ a σ, $\sigma > 0$, které značíme $\mathcal{N}(\mu,\sigma^2)$, je dáno hustotou

$$\varphi_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad x \in \mathcal{R}.$$

Rozdělení $\mathcal{N}(\mu,\sigma^2)$ má střední hodnotu μ a rozptyl σ^2 a distribuční funkci

$$\Phi_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(y-\mu)^2}{2\sigma^2}}\, dy, \quad x \in \mathcal{R}.$$

Normovaným (standardním) normálním rozdělením rozumíme rozdělení $\mathcal{N}(0,1)$, jeho hustotu značíme obvykle jen φ a distribuční funkci Φ. Pokud náhodná veličina Z má normované normální rozdělení, potom veličina $\mu + \sigma Z$ má rozdělení $\mathcal{N}(\mu,\sigma^2)$ a pro distribuční funkce platí

$$\Phi_{\mu,\sigma}(x) = \Phi\left(\frac{x-\mu}{\sigma}\right), \quad x \in \mathcal{R}.$$

Jednorozměrným případem se zabývat příliš nemusíme. Pouze se zmiňme, že se v generátorech zpravidla využívá inverzní transformace, kde se inverze Φ aproximuje funkcí, kterou lze rychle vyčíslit. V *Mathematice* generujeme n čísel z rozdělení $\mathcal{N}(\mu,\sigma^2)$ následovně.

```
n = 4; μ = 0; σ = 2;
z = RandomReal[NormalDistribution[μ, σ], n];

z // TableForm
```

```
-2.30469
2.58733
-0.00800212
-3.05456
```

Pozor! Parametry normální distribuce v *Mathematice* jsou střední hodnota a směrodatná odchylka (nikoliv rozptyl). Podobným způsobem můžeme generovat čísla z celé řady nejpoužívanějších spojitých distribucí, pro generování ze známých diskrétních distribucí používáme místo funkce *RandomReal* funkci *RandomInteger*.

Dále se věnujme trochu podrobněji mnohorozměrnému normálnímu rozdělení. Pro dimenzi $d \in \mathcal{N}$ charakterizujeme d-rozměrné normální rozdělení $\mathcal{N}(\mu,\Sigma)$ s nosičem $\mathcal{R}^d$ pomocí vektoru středních hodnot $\mu \in \mathcal{R}^d$ a kovarianční matice Σ typu $d \times d$. Libovolná kovarianční matice V je vždy symetrická (ekvivalentně $V = V^T$) a pozitivně semidefinitní ($x^T V x \geq 0 \;\forall\, x \in \mathcal{R}^d$). Symetrická matice (má všechna vlastní čísla reálná) je pozitivně semidefinitní (resp. definitní) právě tehdy, když všechna její vlastní čísla jsou nezáporná (resp. kladná). Normální rozdělení $\mathcal{N}(\mu,\Sigma)$ lze na nejnižší úrovni (s respektem jedinné vlastnosti $\mathcal{R}^d = \mathcal{R} \times \ldots \times \mathcal{R}$) chápat jako kartézský součin veličin s jednorozměrným normálním rozdělením, jejichž závislosti jsou dány kovarianční maticí. Hustota rozdělení $\mathcal{N}(\mu,\Sigma)$ je dána vztahem

$$\varphi_{\mu,\Sigma}(x) = \frac{1}{\sqrt{\det\Sigma}\,\sqrt{2\pi}}\, e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}, \quad x \in \mathcal{R}^d.$$

Jestliže náhodný vektor X má rozdělení $\mathcal{N}(\mu,\Sigma)$, pak jeho i-tá složka má distribuci $\mathcal{N}(\mu_i,\Sigma_{i,i})$ a kovariance j-té a i-té složky jest $\Sigma_{j,i}$, tím jsme zdůvodnili použití přívlastku kovarianční matice Σ. Normované (standardní) normální rozdělení $\mathcal{N}(0_d, I_d)$ vzniká kartézským součinem nezávislých jednorozměrných normovaných normálních rozdělení. Jeho hustota pak přejde na jednodušší tvar

$$\varphi(x) = \frac{1}{\sqrt{2\pi}}\, e^{-\frac{1}{2}x^T x}, \quad x \in \mathcal{R}^d.$$

Lineární transormací vektoru s normálním rozdělením dostaneme opět vektor s normálním rozdělením a platí

$$X \sim \mathcal{N}(\mu, \Sigma) \implies A\,X \sim \mathcal{N}\left(A\,\mu, A\,\Sigma\,A^T\right)$$

pro libovolný d-rozměrný vektor μ, kovarianční matici Σ typu $d \times d$ a lineární transformaci danou maticí A typu $k \times d$ pro $k \in \mathcal{N}$. Ještě dodejme, že budeme pracovat jen s maticemi Σ, které jsou pozitivně definitní. V opačném případě by matice Σ neměla plnou hodnost d, matici Σ by nebylo možné invertovat, neexistovala by hustota spojitá vzhledem k Lebesgueově míře (veškerá míra pravděpodobnosti by byla koncentrována po odečtení μ v podprostoru $\mathcal{R}^d$) a museli bychom tedy zvolit vhodnou transformaci a pracovat s jinými souřadnicemi v nižší dimenzi.

Generování dat z normovaného (standardního) normálního d-rozměrného rozdělení se hlouběji věnovat nebudeme. Jenom se opět zmiňme, že mimo inverzních transformací se ke generování v případě $d = 2$ používá například *Boxova-Mullerova metoda*, která negeneruje jednotlivé souřadnice v $\mathcal{R}^2$, ale směr a vzdálenost od vektoru 0_2. Využívá přitom poznatků. uvedených dále. Má-li náhodný vektor X rozdělení $\mathcal{N}(0_2, I_2)$, pak druhá mocnina vzdálenosti $R^2 = X_1^2 + X_2^2$ má exponenciální rozdělení se střední hodnotou 2 a pro pevný poloměr jsou body (X_1, X_2) rozděleny rovnoměrně na kružnici s tímto poloměrem se středem v počátku 0_2.

My budeme samozřejmě opět používat ke generování n dat z $\mathcal{N}(0_d, I_d)$ funkci *RandomReal*.

```
n = 5; d = 3; μ = 0; σ = 2;
v = RandomReal[NormalDistribution[μ, σ], {n, d}];

v // TableForm
```

```
0.836664   -1.38785   2.09386
-0.795612  -1.16367   -0.298511
-1.15246   0.158595   -1.0848
4.33725    -0.074659  2.03593
-1.1275    -1.84722   4.31647
```

Nyní konečně ukažme, jak generovat data z obecného d-rozměrného normálního rozdělení $\mathcal{N}(\mu,\Sigma)$, na základě daných parametrů μ a Σ. Nechť náhodná veličina Z má normální rozdělení $\mathcal{N}(0_d, I_d)$, takovou již dokážeme generovat. Definujeme-li náhodnou veličinu $X = \mu + A\,Z$, pak X má normální rozdělení $\mathcal{N}\left(\mu, A\,A^T\right)$. Ihned vidíme, že náhodná veličina X bude mít požadované rozdělení $\mathcal{N}(\mu,\Sigma)$, pokud najdeme takovou matici A, která splňuje $A\,A^T = \Sigma$. Zápis matice v uvedeném tvaru součinu dvou vzájemně transponovaných matic se nazývá *Choleského rozklad*. Pro pozitivně definitní reálnou matici Σ existuje právě jeden takový rozklad až na změnu znaménka matice A. Postup nalezení

rozkladu je podobně jako například převod matice na trojúhelníkový tvar jednoduchou technickou záležitostí. V *Mathematice* k tomuto účelu voláme funkci *CholeskyDecomposition* (vrací činitel vpravo, tj. matici, kterou jsme značili A^T). Data s malým rozsahem ale můžeme podobně jako v jednorozměrném případě transformovat odmocninou ze Σ využitím maticového mocnění *MatrixPower*. Uveďme příklad generování *n* vektorů z normálního rozdělení $\mathcal{N}(\mu,\Sigma)$.

```
n = 5; Σ = {{3, 2, 0}, {2, 3, -1}, {0, -1, 3}}; d = Length[Σ]; μ = {m1, m2, m3};
v = Map[μ + Transpose[CholeskyDecomposition[Σ]] # & ,
    RandomReal[NormalDistribution[0, 1], {n, d}]];

v // TableForm
```

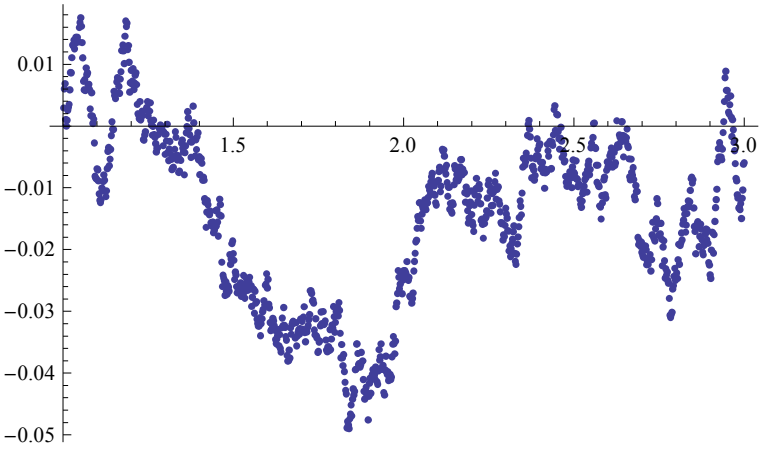
$-0.197736 + m_1$	$1.40694 + m_2$	$-1.3168 + m_3$
$-0.734222 + m_1$	$-0.773171 + m_2$	$0.16198 + m_3$
$4.44132 + m_1$	$0.621918 + m_2$	$2.06648 + m_3$
$-4.40106 + m_1$	$-3.1884 + m_2$	$0.602022 + m_3$
$-2.04264 + m_1$	$1.28558 + m_2$	$-1.95008 + m_3$

Vektor středních hodnot jsem ponechal v obecném tvaru, aby vynikla znaménka korelací mezi složkami.

■ 2.6 Trajektorie Wienerova procesu

Wienerův proces je markovský proces se spojitými trajektoriemi a s nezávislými přírůstky, kde každý přírůstek za dobu Δt má normální rozdělení $\mathcal{N}(0,\sigma^2 \Delta t)$. Zabývat se budeme jednorozměrným případem, zobecnění pro mnohorozměrný případ není obtížné. Pro účely simulací zpravidla stačí kótovat proces jen v diskrétních časových okamžicích, např. v bodech tvořících ekvidistantní dělení intervalu, na kterém je proces definován, a pracovat tak s Markovovými řetězci s podobnými vlastnostmi. Podrobněji se čtenář může seznámit s náhodnými procesy ve skriptech [P07].
V *Mathematice* můžeme generovat jednu trajektorii řetězce poměrně snadno.

```
n = 1000; {a, b} = {1, 3}; Δt = (b - a)/n;
path = Accumulate[RandomReal[NormalDistribution[0, Δt], n]];

ListPlot[{Range[a, b, Δt] // Rest, path} // Transpose]
```



V uvedeném příkladu jsme vlastně realizovali výběr z Markovova řetězce zastupujícího Wienerův proces na intervalu [*a*,*b*]. Ke generování jedné trajektorie jsme museli generovat *n* čísel z normálního rozdělení.

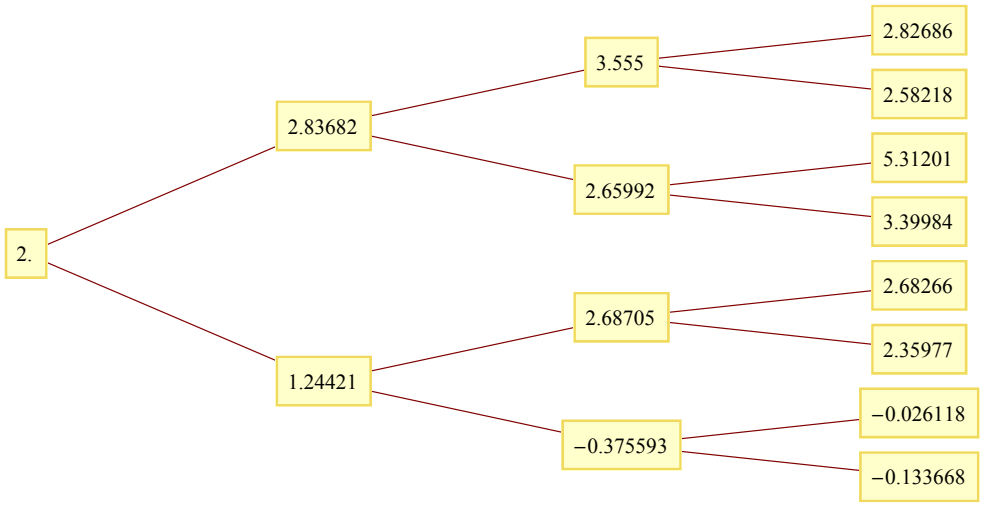
Dále si ukážeme postup, jak generovat strom náhodných dat nahrzující b^m trajektorií, který využijeme v samotném závěru textu při oceňování amerických opcí. Sestrojíme funkci *GenerateNormalTree* generující v každém kroku *m* možných přechodů řetězce najednou. Protože formát dat nebude úplně přehledný, vytvoříme i několik funkcí pro snazší manipulaci se stromy. Pro lepší prohlížení dat sestavíme nejprve funkci *ToTreeData*, která data převede do formy zobrazitelné vestavěnou funkcí *TreePlot*, a dále funkci *TreePart* přistupující k jednotlivým prvkům stromu. Přestože funkce *ToTreeData* ani *TreePart* dále nepoužijeme, může si na nich čtenář rozšířit (příp. procvičit) znalosti rekurzivního procházení stromů. Podobné postupy ještě v poslední kapitoleme potřebovat budeme.

```
Remove[GenerateNormalTree, ToTreeData, TreePart]

GenerateNormalTree[m_, b_, μ_, σ_, S_] :=
 Nest[
  Map[
    {#, RandomReal[NormalDistribution[# + μ, σ], b]} &,
    #,
    {Depth[#] - 1}] &,
  S // N, m]

ToTreeData[r_Real, M_?MatrixQ] :=
 {Map[Rule[r, #] &, Evaluate[M[[All,1,1]]], M} // Flatten

ToTreeData[r_Real, v_?(VectorQ[#, NumericQ] &)] :=
 Map[Rule[r, #] &, v]

ToTreeData[Data_] :=
 Module[{r, L}, Data //. {r_Real, L_List} → ToTreeData[r, L]]

TreePart[Tree_, Pos_List] :=
 Module[
  {expr = Apply[Part,
     Prepend[Riffle[Pos, 2, {1, -2, 2}], Tree]
    ]},
  If[Head[expr] === List, First[expr], expr]]
```

Nyní použitím deklarovaných funkcí vygenerujeme strom zastupující b^m různých průchodů Markovovým řetězcem s počátečním stavem *S*, kde jednotlivé přechody řetězce budou respektovat rozdělení $\mathcal{N}(\mu, \sigma^2 \Delta t)$. Výsledná data zobrazíme nejprve přímo tabulkou, potom funkcí *TreePlot* a nakonec ukážeme, jak přistupovat k prvkům stromu pomocí sekvence charakterizující větvení.

```
m = 3; b = 2; μ = 0; Δt = 1; S = 2;
Data = GenerateNormalTree[m, b, μ, Δt, S];

Data // TableForm
Data // ToTreeData // TreePlot[#,
    Left, VertexLabeling → True, AspectRatio → 1 / 2] &

{
  TreePart[Data, {}],
  TreePart[Data, {1}],
  TreePart[Data, {1, 1}],
  TreePart[Data, {1, 1, 2}]
 } // TableForm
```

```
2.
1.24421                                   2.83682
-0.375593            2.68705              2.65992          3.555
-0.133668  -0.026118  2.35977  2.68266    3.39984  5.31201  2.58218  2.82686
```




```
2.
1.24421
-0.375593
-0.026118
```

Při porovnání sekvencí s obrázkem vidíme, že *i*-tý prvek sekvence odpovídá *i*-té hladině (hloubce) větví stromu a jednotlivé *b*-tice větví jsou číslovány v obrázku zdola.

3 Metody Monte Carlo

Jak již bylo naznačeno v úvodu, *metodou Monte Carlo (MC) budeme rozumět takovou simulační metodu, která pracuje s pravděpodobnostními (stochastickými) modely, v nichž náhodné výběry nahrazuje vhodně volenými daty*. Zde máme vhodně volenými daty namysli náhodně generovaná data z požadovaného teoretického rozdělení, nebo přesněji pseudonáhodnou realizaci náhodného výběru potřebného pro teoretické řešení úlohy. Z hlediska teorie pravděpodobnosti lze každou simulační úlohu přeformulovat tak, abychom si vystačili pouze s rovnoměrnými rozděleními na intervalu [0,1], jinými slovy můžeme všechny naše předpoklady o neurčitosti vyjádřit právě těmito rozděleními.

V praxi takto často postupujeme, ale existují i v jistém smyslu optimálnější postupy. Složitými transformacemi pseudonáhodných dat se totiž mohou například jen vlivem zaokrouhlování vytrácet některé významné vlastnosti pseudonáhodných čísel, jako nekorelovanost s jinými daty, na kterých by vůbec neměla záviset, nebo správné četnosti výskytu na jistých množinách apod. V numerické matematice v této souvislosti hovoříme o *diskretizačních chybách* vznikajících přechodem k diskrétním modelům a o *suboptimalitě* řešení. Tyto problémy ale mají spíše technický charakter a přesahují rámec této práce.

3.1 Obyčejná metoda Monte Carlo

Pod pojmem *obyčejná metoda Monte Carlo* si budeme vždy představovat takovou metodu MC, která se nejsnáze implementuje a není nijak optimalizovaná. Při srovnávání metod MC pro nás bude dobrým etalonem. Podívejme se, jak můžeme takovou metodou odhadovat jednorozměrný integrál nebo velikosti množin. Upozorněme, že pro výpočet jednorozměrných integrálů existují i efektivnější postupy, přesto nám úloha postačí pro demonstrování základních principů. Výhody metody Monte Carlo vyniknou až při výpočtu mnohorozměrných integrálů, a to díky rychlosti konvergence odhadu $O(n^{-2})$, která obecně nezávisí na dimenzi.

3.2 Odhad integrálu

Nechť je dán interval $[a,b]$, $a, b \in \mathcal{R}$, a nechť $f \in \mathcal{L}_2([a,b])$. Chceme odhadnout integrál

$$I = \int_a^b f(x)\, \mathrm{d}x.$$

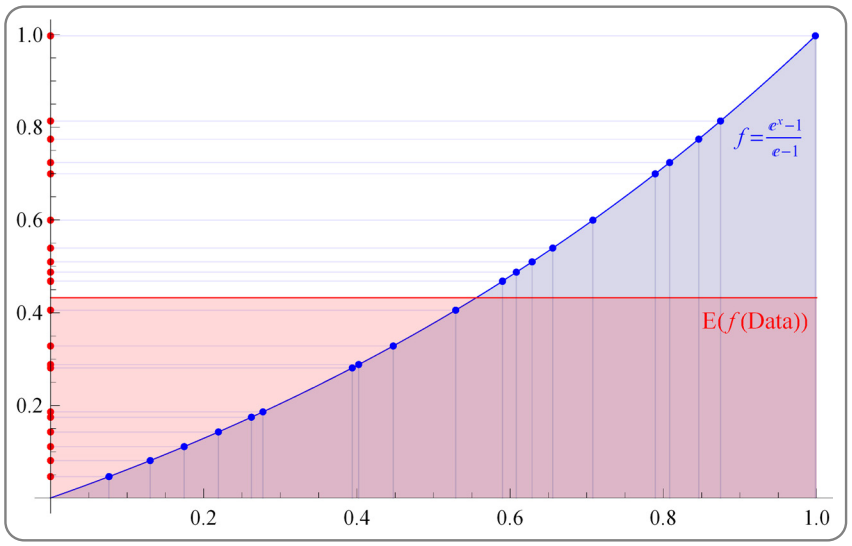
Pokud pro $n \in \mathcal{N}$ bude $\mathbb{X}_n = \{X_1, \ldots, X_n\}$ náhodný výběr z rovnoměrného rozdělení na $[a,b]$ (s hustotou $(b-a)^{-1}\, \chi_{(b-a)}$ vzhledem k λ), potom odhad

$$\hat{I}_n = \frac{b-a}{n} \sum_{i=1}^{n} f(X_i)$$

je nestranným odhadem I, neboť

$$\begin{aligned} \mathrm{E}\, \hat{I}_n &= \mathrm{E}\, \frac{b-a}{n} \sum_{i=1}^{n} f(X_i) = \frac{b-a}{n} \sum_{i=1}^{n} \mathrm{E}\, f(X_i) = (b-a)\, \mathrm{E}\, f(X_1) = \\ &= (b-a) \int_{\mathcal{R}} f(x)\, \frac{1}{b-a}\, \chi_{(b-a)}(x)\, \mathrm{d}x = (b-a) \int_a^b f(x)\, \frac{1}{b-a}\, \mathrm{d}x = I. \end{aligned}$$

Pro malé rozsahy dat n můžeme ilustrovat situaci obrázkem. Obsah modré plochy pod grafem funkce f odhadujeme obsahem červeného obdélníku.



Obr. 3.1. Odhad plochy pod grafem funkce f.

Teoretický rozptyl odhadu (*chyba odhadu*) jest

$$\begin{aligned}
\operatorname{var} \hat{I}_n &= \operatorname{var} \tfrac{b-a}{n} \textstyle\sum_{i=1}^n f(X_i) \\
&= \tfrac{(b-a)^2}{n^2} \operatorname{var} \textstyle\sum_{i=1}^n f(X_i) \\
&= \tfrac{(b-a)^2}{n^2} \left(\textstyle\sum_{i=1}^n \operatorname{var} f(X_i) + \sum_{i=1}^n \sum_{j=1, j\neq i}^n \operatorname{cov}\left(f(X_i), f\left(X_j\right)\right)\right) \\
&= \tfrac{(b-a)^2}{n} \operatorname{var} f(X_1) \\
&= \tfrac{(b-a)^2}{n} \int_R (f(x) - I)^2 \, \tfrac{1}{b-a} \chi_{(b-a)}(x) \, \mathrm{d}x \\
&= \tfrac{b-a}{n} \int_a^b (f(x) - I)^2 \, \mathrm{d}x.
\end{aligned}$$

Na tomto místě si uvědomme, že $\hat{I}_n$ vlastně není ničím jiným, než výběrovým průměrem, a tedy nestranným a konzistentním odhadem střední hodnoty náhodné veličiny $(b-a)\, f(X_1)$. Rozptyl této náhodné veličiny můžeme odhadnout jeho výběrovým protějškem

$$S_n^2 = \tfrac{1}{n-1} \textstyle\sum_{i=1}^n \left((b-a)\, f(X_i) - \hat{I}_n \right)^2.$$

Při numerických výpočtech ale raději používáme následující výhodnější ekvivalentní zápis výběrového rozptylu

$$\begin{aligned}
S_n^2 &= \tfrac{1}{n-1} \textstyle\sum_{i=1}^n \left((b-a)\, f(X_i) - \hat{I}_n \right)^2 \\
&= \tfrac{1}{n-1} \left(\textstyle\sum_{i=1}^n \left((b-a)^2 f^2(X_i) + \hat{I}_n^{\,2} \right) - \sum_{i=1}^n 2\,(b-a)\, f(X_i)\, \hat{I}_n \right) \\
&= \tfrac{1}{n-1} \left(\textstyle\sum_{i=1}^n \left((b-a)^2 f^2(X_i) + \hat{I}_n^{\,2} \right) - 2 \sum_{i=1}^n (b-a)\, f(X_i)\, \tfrac{b-a}{n} \sum_{j=1}^n f\left(X_j\right) \right) \\
&= \tfrac{1}{n-1} \left(\textstyle\sum_{i=1}^n \left((b-a)^2 f^2(X_i) + \hat{I}_n^{\,2} \right) - 2 \sum_{k=1}^n \tfrac{(b-a)}{n} \sum_{i=1}^n f(X_i) \quad \tfrac{b-a}{n} \sum_{j=1}^n f\left(X_j\right) \right) \\
&= \tfrac{1}{n-1} \left(\textstyle\sum_{i=1}^n \left((b-a)^2 f^2(X_i) + \hat{I}_n^{\,2} \right) - 2 \sum_{k=1}^n \hat{I}_n^{\,2} \right) =
\end{aligned}$$

$$= \frac{1}{n-1}\left(\sum_{i=1}^{n}(b-a)^2 f^2(X_i) + \sum_{i=1}^{n}\hat{I}_n^{\,2} - 2\sum_{k=1}^{n}\hat{I}_n^{\,2}\right)$$
$$= \frac{(b-a)^2}{n-1}\sum_{i=1}^{n} f^2(X_i) - \frac{n}{n-1}\hat{I}_n^{\,2}.$$

Teoretický rozptyl odhadu $\hat{I}_n$ zpravidla aproximujeme právě s využitím výběrového rozptylu jako

$$\begin{aligned}\operatorname{var}\hat{I}_n &= \operatorname{var}\frac{b-a}{n}\sum_{i=1}^{n} f(X_i)\\ &= \frac{1}{n^2}\operatorname{var}(b-a)\sum_{i=1}^{n} f(X_i)\\ &= \frac{1}{n}\operatorname{var}(b-a) f(X_1)\\ &\approx \frac{1}{n}S_n^2\\ &= \frac{(b-a)^2}{n(n-1)}\sum_{i=1}^{n} f^2(X_i) - \frac{1}{n-1}\hat{I}_n^{\,2}.\end{aligned}$$

Je-li $\operatorname{var}(b-a) f(X_1) > 0$, pak přímou aplikací centrální limitní věty dostáváme, že náhodná veličina

$$\sqrt{n}\,\frac{\frac{1}{n}\sum_{i=1}^{n}(b-a) f(X_i) - I}{\sqrt{\operatorname{var}(b-a) f(X_1)}}$$

má asymptoticky normované normální rozdělení $\mathcal{N}(0,1)$. Protože výběrový rozptyl je konzistentním (i nestranným) odhadem rozptylu, konverguje podíl $S_n^2 \,/\, \operatorname{var}(b-a) f(X_1) \to 1$ pro $n \to \infty$ skoro jistě, tudíž také náhodná veličina

$$\sqrt{n}\,\frac{\frac{1}{n}\sum_{i=1}^{n}(b-a) f(X_i) - I}{\sqrt{S_n^2}}$$

má v limitě rozdělení $\mathcal{N}(0,1)$. Odtud už se snadno odvodí intervalový odhad pro I na hladině α

$$\left(\hat{I}_n - \Phi^{-1}_{1-\alpha/2}\,\frac{S_n}{\sqrt{n}},\ \hat{I}_n + \Phi^{-1}_{1-\alpha/2}\,\frac{S_n}{\sqrt{n}}\right),$$

kde Φ^{-1} je kvantil rozdělení $\mathcal{N}(0,1)$. K numerickému dopočtení úlohy stačí do vztahu pro odhad dosazovat místo náhodného výběru $\mathbb{X}_n = \{X_1, \ldots, X_n\}$ jeho pseudonáhodnou realizaci $\mathbb{x}_n = \{x_1, \ldots, x_n\}$. Přejděme k vlastní implementaci metody v programu *Mathematica*. Funkci *MCM* odhadující integrál budeme definovat tak, že jejím posledním argumentem mohou být buďto již požadovaná data, nebo jen rozsah dat, která se vytvoří při volání funkce.

```
Remove[MCM]

MCM[f_, n_Integer] :=
 MCM[f, {0, 1}, RandomReal[{0, 1}, n]] /; n > 0

MCM[f_, {a_, b_}, n_Integer] :=
 MCM[f, {a, b}, RandomReal[{a, b}, n]] /; n > 0

MCM[f_, {a_, b_}, Data_List] := (b - a) Mean[Map[f, Data]]
```

Funkce počítající rozptyl a interval spolehlivosti budeme zatím vytvářet pro větší přehlednost odděleně, přestože prakticky by samozřejmě byl takový postup získávání všech informací současně neefektivní. Nejprve ukážeme, jak bychom odhady zapsali téměř přímým přepisem odvozevých vztahů.

```
Remove[VarMCM, CIMCM]

VarMCM[f_, {a_, b_}, Data_List] := Module[
  {n = Length[Data],
   I = (b - a) Mean[Map[f, Data]]},
  (b - a)^2 / (n (n - 1)) Total[Map[(f[#])^2 &, Data]] - 1 / (n - 1) I^2]

CIMCM[f_, {a_, b_}, Data_List, ConfLevel_] := Module[
  {n = Length[Data],
   Q = Quantile[NormalDistribution[0, 1], (ConfLevel + 1) / 2],
   I = (b - a) Mean[Map[f, Data]],
   S2}, S2 = (b - a)^2 / (n - 1) Total[Map[(f[#])^2 &, Data]] - n / (n - 1) I^2;
  {I - Q Sqrt[S2] / Sqrt[n], I + Q Sqrt[S2] / Sqrt[n]}]
```

Jak se ale za chvíli přesvědčíme, daleko lepší je využívat vestavěných funkcí *Mathematicy*. Funkce *Variance* aplikovaná na seznam počítá výběrový rozptyl a podobně *MeanCI* (obsažená v balíčku funkcí *HypothesisTesting* nenačítajícím se automaticky po spuštění programu) počítá interval spolehlivosti ekvivalentně jako funkce výše, jen místo Q používá kvantil Studentova t-rozdělení s $n-1$ stupni volnosti, takže výsledný interval spolehlivosti je nepatrně širší. Při implementaci dalších metod raději upřednostníme tyto funkce.

```
Remove[MathVarMCM, MathCIMCM]
Needs["HypothesisTesting`"]

MathVarMCM[f_, {a_, b_}, Data_List] :=
 Variance[Map[(b - a) f, Data]] / Length[Data]

MathCIMCM[f_, {a_, b_}, Data_List, ConfLevel_] :=
 MeanCI[Map[(b - a) f, Data], ConfidenceLevel → ConfLevel]
```

Nyní metodu aplikujeme na výpočet integrálu z funkce f definované na intervalu [0,1] předpisem

$$f(x) = \frac{e^x - 1}{e - 1}.$$

Využijeme přitom generování dat s rozsahem $n = 10\,000$. Na výstupu dostaneme nejprve dvojici $\{t, \hat{I}_n\}$, kde t udává čas (ve vteřinách) potřebný k výpočtu $\hat{I}_n$, což je odhad integrálu v souladu se značením zavedeným výše. Na dalších dvou řádcích následují aproximace rozptylu tohoto odhadu počítané dvěma způsoby, opět ve tvaru $\{t$, odhad *var* $\hat{I}_n\}$. Dále porovnáme různě spočítané intervaly spolehlivosti pro $\hat{I}_n$ na úrovni spolehlivosti 0.95 z řádků v podobě $\{t, \{c_l, c_u\}\}$. Poslední výstup slouží jen pro srovnání, půjde o numericky vyčíslenou skutečnou hodnotu integrálu I.

```
Remove[f, x]

{a, b} = {0, 1}; f[x_] := (e^x - 1)/(e - 1)
Data = RandomReal[{a, b}, 10 000];

Timing[MCM[f, {a, b}, Data]]

Timing[VarMCM[f, {a, b}, Data]]
Timing[MathVarMCM[f, {a, b}, Data]]

Timing[CIMCM[f, {a, b}, Data, 0.95]]
Timing[MathCIMCM[f, {a, b}, Data, 0.95]]

∫_a^b f[x] dx // N
```

```
{0.093, 0.419987}

{0.204, 8.26072 × 10^-6}

{0.109, 8.26072 × 10^-6}

{0.203, {0.414353, 0.42562}}

{0.093, {0.414353, 0.425621}}

0.418023
```

Jak je vidět, funkce *MathVarMCM* a *MathCIMCM* počítají odhady ekvivalentní odhadům *VarMCM* a *CIMCM* (vzpomeňme ale použití jiného předpokládaného rozdělení odhadu). Navíc pracují přibližně dvakrát rychleji.

■ 3.3 Odhad objemu množiny

Úlohu odhadu objemu množiny lze považovat za restrikci úlohy odhadu integrálu, kde za funkci f volíme indikátor množiny, jejíž míru odhadujeme. Tento speciální případ můžeme uvést v obecnějším kontextu, totiž nezávisle na dimenzi množiny. Předpokládejme, že množina M, jejíž objem chceme odhadnout, je obsažena v nějaké množině $\Omega \supset M$ s předem známým konečným objemem a navíc takové, že jsme schopni realizovat náhodný výběr $\mathbb{U}_n = \{U_1, \ldots, U_n\}$ z rovnoměrného rozdělení na Ω. Pokud aplikujeme na náhodný výběr (po prvcích) indikátor množiny M, dostaneme jiný náhodný výběr $\mathbb{Y}_n = \{Y_1, \ldots, Y_n\}$, přičemž každá z veličin výběru $\mathbb{Y}_n$ má nyní alternativní rozdělení s parametrem p, který neudává nic jiného, než poměr objemů M a Ω, neboť právě tento poměr odpovídá pravděpodobnosti, že náhodně vybraný bod z Ω je zároveň z M.

Za *odhad λ(M) objemu množiny M metodou Monte Carlo* potom prohlásíme součin

$$\hat{\lambda}(M)_n = \lambda(\Omega)\,\frac{1}{n}\sum_{i=1}^{n} Y_i = \lambda(\Omega)\,\frac{1}{n}\sum_{i=1}^{n} \chi_M(U_i),$$
$$Y_i \sim \mathcal{Alt}(p)\,, \quad p = \frac{\lambda(M)}{\lambda(\Omega)}.$$

Zabývat se nestranností nebo konzistencí má smysl až při zavedení nějaké normy (a konvergence v této normě). Obě tyto vlastnosti ani nemusíme ověřovat, pokud pracujeme s obvyklým lineárním prostorem $\mathcal{R}^n \supset \Omega$, neboť stačí opět říct, že používáme výběrový průměr. Nabízí se ale otázka, jestli je výhodnější odhadovat oblast pod grafem jednorozměrné funkce jako integrál, nebo jako míru dvourozměrné množiny. Generování vícerozměrných dat má samozřejmě větší časovou náročnost, ale na druhou stranu lze v některých případech přeformulovat úlohu tak, že stačí vyhodnocovat nerovnosti namísto vyčíslování funkčních hodnot. Uveďme nejprve obecnou implementaci výpočtu.

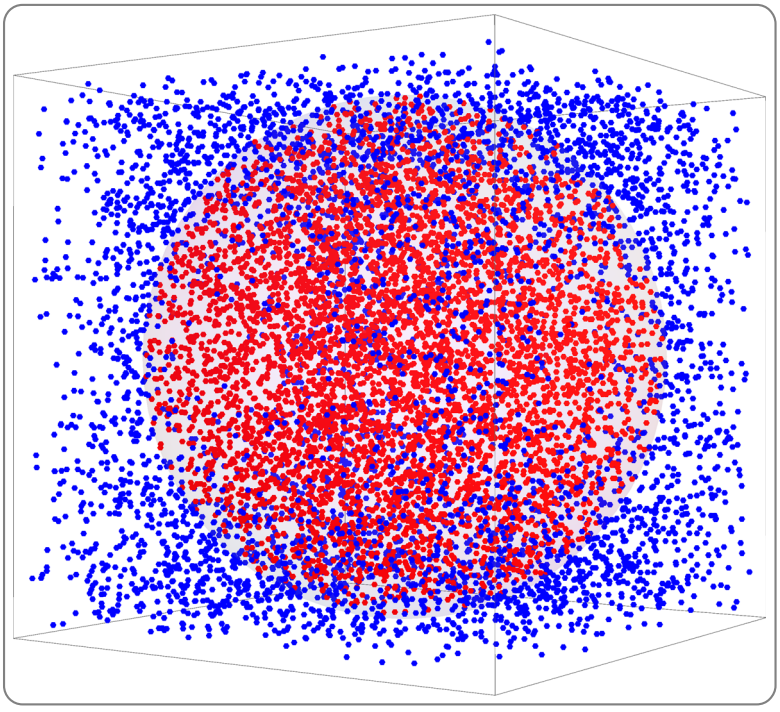
```
Remove[MCM]

MCM[λΩ_, CritM_, Data_List] :=
 Ω Mean[Map[Boole[CritM[#]] &, Data]] // N
```

Nyní například uvažujme pro jednoduchost úlohu odhadu objemu jednotkové sféry v $\mathcal{R}^3$. Celá sféra (označme množinu bodů uvnitř sféry M) leží ve zobecněném intervalu [-1,1]×[-1,1]×[-1,1], který si zvolíme jako množinu Ω. Snadno se nahlédne, že $\lambda(\Omega) = 8$. Kritérium, které nám určí, jestli náhodně zvolený bod (vektor) z Ω náleží také M je v tomto případě nasnadě. Jedná se pouze o srovnání jeho eukleidovské normy s poloměrem sféry 1. Za výpočtem ještě následuje ilustrace pro 10 000 bodů. Body uvnitř sféry jsou vykresleny červeně, vnější modře.

```
Ω = 8; Data = RandomReal[{-1, 1}, {10 000, 3}];
CritM = (EuclideanDistance[{0, 0, 0}, #] ≤ 1) &;
MCM[8, CritM, Data]
```

$$\frac{4}{3}\pi \text{ // N}$$

```
4.248

4.18879
```



Obr. 3.2. Odhad objemu jednotkové sféry.

4 Techniky redukce rozptylu

V této kapitole si ukážeme, jak je možné obecnou metodu Monte Carlo vylepšit. Hlavní principy budeme znovu demonstrovat na úloze výpočtu jednorozměrného integrálu na konečném intervalu. Není ale obtížné dále uvedené postupy zobecnit (až na metodu antitetických proměnných). Zvyšování přesnosti výpočtu je v zásadě možné na základě apriorní znalosti nějaké vlastnosti vyšetřované funkce. V následujících několika podkapitolách budeme opět předpokládat, že máme dán interval $[a,b]$, $a, b \in \mathcal{R}$, $f \in \mathcal{L}_2([a,b])$ a chceme odhadnout integrál I z funkce f přes interval $[a,b]$.

■ 4.1 Korelovaný výběr

Korelovaný výběr (metodu řídících proměnných) využijeme v případě, že máme dobrou představu o průběhu funkce f, z níž chceme integrál určit. Odhadneme-li aspoň velice hrubě funkci f takovou funkcí $g \in \mathcal{L}_2([a,b])$, ze které dovedeme spočítat integrál snadno, potom obyčejnou metodou MC odhadujeme integrál z rozdílu funkcí $f - g$. Rozptyl výsledného výstupu zde vlastně redukujeme snížením rozptylu vstupních dat. K teoretickému řešení znovu využijeme náhodného výběru $\mathbb{X}_n = \{X_1, \ldots, X_n\}$ z rovnoměrného rozdělení $\mathcal{U}([a,b])$. Pokud definujeme

$$\begin{aligned} I &= \int_a^b f(x)\,dx, \\ I_1 &= \int_a^b g(x)\,dx, \\ I_2 &= \int_a^b (f(x) - g(x))\,dx, \end{aligned}$$

pak $I = I_1 + I_2$, hodnotu I_1 považujeme za známou a tedy I můžeme odhadovat jako

$$\hat{I} = I_1 + \hat{I}_2 = \int_a^b g(x)\,dx + \frac{b-a}{n} \sum_{i=1}^n (f(X_i) - g(X_i)).$$

Tento odhad je zřejmě nestranný, neboť odhad $\hat{I}_2$ je nestranný, jak jsme ukázali u obecné metody. Rozptyl odhadu vypadá následovně

$$\begin{aligned} \operatorname{var} \hat{I} &= \operatorname{var} \hat{I}_2 \\ &= \frac{(b-a)^2}{n^2} \operatorname{var} \sum_{i=1}^n (f(X_i) - g(X_i)) \\ &= \frac{(b-a)^2}{n^2} \left(\sum_{i=1}^n \operatorname{var} (f(X_i) - g(X_i)) + \sum_{i=1}^n \sum_{j=1, j\neq i}^n \operatorname{cov} ((f-g)(X_i), (f-g)(X_i)) \right) \\ &= \frac{(b-a)^2}{n^2} \sum_{i=1}^n \operatorname{var} (f(X_i) - g(X_i)) \\ &= \frac{(b-a)^2}{n} (\operatorname{var} f(X_1) - 2 \operatorname{cov}(f(X_1), g(X_1)) + \operatorname{var} g(X_1)). \end{aligned}$$

Kdybychom předpokládali, že rozptyl $f(X_1)$ je přibližně stejný jako rozptyl $g(X_1)$, můžeme dále psát

$$\begin{aligned} \operatorname{var} \hat{I} &= \frac{(b-a)^2}{n} \sqrt{\operatorname{var} f(X_1) \operatorname{var} g(X_1)} \left(\frac{\operatorname{var} f(X_1) + \operatorname{var} g(X_1)}{\sqrt{\operatorname{var} f(X_1) \operatorname{var} g(X_1)}} - 2 \frac{\operatorname{cov}(f(X_1), g(X_1))}{\sqrt{\operatorname{var} f(X_1) \operatorname{var} g(X_1)}} \right) \\ &\approx \frac{(b-a)^2}{n} 2 \sqrt{\operatorname{var} f(X_1) \operatorname{var} g(X_1)} \; (1 - \operatorname{corr} (f(X_1), g(X_1))). \end{aligned}$$

Rozptyl odhadu by pak závisel hlavně na tom, jak výrazně jsou korelované veličiny $f(X_1)$ a $g(X_1)$.

Povšimněme si, že další výpočty týkající se rozptylu již nemusíme odvozovat. Ve výsledných vztazích pro obyčejnou metodu MC stačí psát místofunkce f rozdíl $f-g$. Potom ihned dostáváme, že výběrový rozptyl, a tedy nestranný a konzistentní odhad rozptylu veličiny $(b-a)\,(f-g)\,(X_1)$, jest

$$\begin{aligned} S^2 &= \tfrac{1}{n-1}\sum_{i=1}^{n}\left((b-a)\,(f-g)\,(X_i)-\hat{I}\right)^2 \\ &= \tfrac{(b-a)^2}{n-1}\sum_{i=1}^{n}(f-g)^2\,(X_i)-\tfrac{n}{n-1}\,\hat{I}^2. \end{aligned}$$

Stejně tak dostaneme odhad rozptylu odhadu $\hat{I}_n$

$$\begin{aligned} \operatorname{var}\hat{I}_n &= \operatorname{var}\tfrac{b-a}{n}\sum_{i=1}^{n}(f-g)\,(X_i) \\ &= \tfrac{1}{n}\operatorname{var}(b-a)\,f(X_1) \\ &\approx \tfrac{1}{n}\,S^2 \\ &= \tfrac{(b-a)^2}{n(n-1)}\sum_{i=1}^{n}(f-g)^2\,(X_i)-\tfrac{1}{n-1}\,\hat{I}^2 \end{aligned}$$

a intervalový odhad pro I na hladině α

$$\left(\hat{I}-\Phi^{-1}_{1-\alpha/2}\,\frac{S}{\sqrt{n}},\ \hat{I}+\Phi^{-1}_{1-\alpha/2}\,\frac{S}{\sqrt{n}}\right),$$

Také implementace vypadá podobně jako v obecné metodě Monte Carlo, na vstupu přibude funkce g.

```
Remove[MCM, VarMCM, CIMCM]
Needs["HypothesisTesting`"]

MCM[f_, g_, n_Integer] :=
 MCM[f, g, {0, 1}, RandomReal[{0, 1}, n]] /; n > 0

MCM[f_, g_, {a_, b_}, n_Integer] :=
 MCM[f, g, {a, b}, RandomReal[{a, b}, n]] /; n > 0

MCM[f_, g_, {a_, b_}, Data_List] :=
  ∫_a^b g[x] dx + (b - a) Mean[Map[(f[#] - g[#]) &, Data]]

VarMCM[f_, g_, {a_, b_}, Data_List] :=
 Variance[Map[(b - a) (f[#] - g[#]) &, Data]]
 ──────────────────────────────────────────
                Length[Data]

CIMCM[f_, g_, {a_, b_}, Data_List, CL_] := ∫_a^b g[x] dx +
  MeanCI[Map[(b - a) (f[#] - g[#]) &, Data], ConfidenceLevel → CL]
```

Vraťme se k předchozímu příkladu, kde jsme počítali integrál přes $[a,b]$ z funkce

$$f(x) = \frac{e^x - 1}{e - 1}.$$

a volme $g(x) = x$. Rozsah dat $n = 10\,000$ ponecháme jako dříve. Jednotlivé výstupy nyní budou dvojice času potřebného k výpočtu a odhadu integrálu $\{t, \hat{I}_n\}$, dále odhad rozptylu odhadu integrálu, interval spolehlivosti $\{c_l, c_u\}$ pro odhad $\hat{I}_n$ a nakonec vyčíslená skutečná hodnota integrálu I.

```
Remove[f, x]

{a, b} = {0, 1}; f[x_] := (e^x - 1)/(e - 1); g[x_] := x

Data = RandomReal[{a, b}, 10 000];

Timing[MCM[f, g, {a, b}, Data]]
VarMCM[f, g, {a, b}, Data]
CIMCM[f, g, {a, b}, Data, 0.95]
∫_a^b f[x] dx // N
```

```
{0.14, 0.417546}
1.33985 × 10^-7
{0.416828, 0.418263}
0.418023
```

Porovnáním s obecnou metodou MC na první pohled vidíme, že k výpočtu odhadu je sice zapotřebí přibližně o polovinu delší doba, ale rozptyl odhadu se v tomto příkladě sníží zhruba 60-krát.

■ 4.2 Výběr podle důležitosti

Tento postup se zakládá na tom, že v oblastech, kde je veličina $f(X_1)$ málo rozptýlená (tj. v místech, kde je f spíše konstantní), nepotřebujeme provádět více simulací a naopak oblasti, kde má f výrazně různé hodnoty, jsou pro nás důležitější. Namísto náhodného výběru z rovnoměrného rozdělení, který jsme zatím využívali v předchozích úlohách, použijeme za tímto účelem obecněji výběr z rozdělení daného hustotou ρ, kde $\rho \in \mathcal{L}_2([a,b])$ splňuje $\rho \neq 0$ právě tehdy, když $f \neq 0$, a jde o hustotu pravděpodobnostního rozdělení s nosičem $[a,b]$, tedy

$$\rho(x) \geq 0 \;\; \forall\, x \in [a,b] \;\text{ a }\; \int_a^b \rho(x)\, dx = 1.$$

Označíme-li nyní

$$h(x) = \frac{f(x)}{\rho(x)},$$

můžeme vyjádřit hledaný integrál jako

$$I = \int_a^b f(x)\, dx = \int_a^b \frac{f(x)}{\rho(x)}\, \rho(x)\, dx = \int_a^b h(x)\, \rho(x)\, dx$$

a jeho odhad počítat podobně jako u obyčejné metody Monte Carlo s tím rozdílem, že využijeme náhodný výběr $\mathbb{X}_n = \{X_1, \dots, X_n\}$ pocházející z rozdělení s hustotou ρ.

Odhad integrálu

$$\hat{I}_n = \frac{1}{n} \sum_{i=1}^{n} h(X_i)$$

je nestranným odhadem I, protože platí

$$\begin{aligned}
\mathrm{E}\, \hat{I}_n &= \mathrm{E}\, \frac{1}{n} \sum_{i=1}^{n} h(X_i) \\
&= \frac{1}{n} \sum_{i=1}^{n} \mathrm{E}\, h(X_i) \\
&= \mathrm{E}\, h(X_1) \\
&= \int_R h(x)\, \rho(x)\, \chi_{(b-a)}(x)\, \mathrm{d}x \\
&= \int_a^b \frac{f(x)}{\rho(x)}\, \rho(x)\, \mathrm{d}x \\
&= \int_a^b f(x)\, \mathrm{d}x = I.
\end{aligned}$$

Vidíme, že odhad není přímo závislý na volbě intervalu, tuto informaci v sobě nese hustota ρ, a tedy funkce h. Teoretický rozptyl odhadu jest

$$\begin{aligned}
\mathrm{var}\, \hat{I}_n &= \mathrm{var}\, \frac{1}{n} \sum_{i=1}^{n} h(X_i) \\
&= \frac{1}{n^2}\, \mathrm{var} \sum_{i=1}^{n} h(X_i) \\
&= \frac{1}{n^2} \left(\sum_{i=1}^{n} \mathrm{var}\, h(X_i) + \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} \mathrm{cov}\left(h(X_i), h\left(X_j\right)\right)\right) \\
&= \frac{1}{n}\, \mathrm{var}\, h(X_1) \\
&= \frac{1}{n} \int_R (h(x) - I)^2\, \rho(x)\, \chi_{(b-a)}(x)\, \mathrm{d}x \\
&= \frac{1}{n} \int_a^b (h(x) - I)^2\, \rho(x)\, \mathrm{d}x.
\end{aligned}$$

Rozptyl $\mathrm{var}\, h(X_1)$ odhadujeme výběrovým rozptylem (vzorec se odvodí analogickým postupem jako u obyčejné metody Monte Carlo)

$$\begin{aligned}
S_n^2 &= \frac{1}{n-1} \sum_{i=1}^{n} \left(h(X_i) - \hat{I}_n\right)^2 \\
&= \frac{1}{n-1} \sum_{i=1}^{n} h^2(X_i) - \frac{n}{n-1}\, \hat{I}_n^{\,2},
\end{aligned}$$

a tedy přibližně (asymptoticky) platí

$$\begin{aligned}
\mathrm{var}\, \hat{I}_n &= \frac{1}{n}\, \mathrm{var}\, h(X_1) \\
&\approx \frac{1}{n}\, S_n^2 \\
&= \frac{1}{n(n-1)} \sum_{i=1}^{n} h^2(X_i) - \frac{1}{n-1}\, \hat{I}_n^{\,2}.
\end{aligned}$$

Z centrální limitní věty dostáváme, že v limitě pro $n \to \infty$ se náhodná veličina

$$\sqrt{n}\, \frac{\frac{1}{n} \sum_{i=1}^{n} h(X_i) - I}{\sqrt{\mathrm{var}\, h(X_1)}}$$

a také veličina

$$\sqrt{n}\ \frac{\frac{1}{n}\sum_{i=1}^{n} h(X_i) - I}{\sqrt{S_n^2}}$$

řídí normovaným normálním rozdělením $\mathcal{N}(0,1)$, mají-li výrazy smysl (pro každé n). Proto intervalovým odhadem pro I na hladině α je

$$\left(\hat{I}_n - \Phi^{-1}_{1-\alpha/2}\ \frac{S_n}{\sqrt{n}},\ \hat{I}_n + \Phi^{-1}_{1-\alpha/2}\ \frac{S_n}{\sqrt{n}}\right),$$

kde Φ^{-1} je kvantil rozdělení $\mathcal{N}(0,1)$. Uvedené úvahy byly podrobněji rozepsány u obyčejné metody Monte Carlo, kde jsme postupovali zcela analogicky. Nyní se podívejme, jaká je nejoptimálnější volba hustoty ρ. Z výpočtu rozptylu odhadu integrálu je patrné, že chceme minimalizovat rozptyl $\operatorname{var} h(X_1)$, který můžeme vyjádřit také jako

$$\begin{aligned}
\operatorname{var} h(X_1) &= \int_a^b (h(x) - I)^2\, \rho(x)\, dx \\
&= \int_a^b \left(h^2(x)\, \rho(x) - 2\, h(x)\, \mathrm{I}\rho(x) + I^2\, \rho(x)\right) dx \\
&= \int_a^b \left(\frac{f^2(x)}{\rho^2(x)}\, \rho(x) - 2\, \frac{f(x)}{\rho(x)} \left(\int_a^b f(y)\, dy\right) \rho(x) + \left(\int_a^b f(y)\, dy\right)^2 \rho(x)\right) dx \\
&= \int_a^b \frac{f^2(x)}{\rho(x)}\, dx\ - 2 \int_a^b f(x) \left(\int_a^b f(y)\, dy\right) dx + \int_a^b \left(\int_a^b f(y)\, dy\right)^2 \rho(x)\, dx \\
&= \int_a^b \frac{f^2(x)}{\rho(x)}\, dx\ - 2 \left(\int_a^b f\ (y)\, dy\right)^2 + \left(\int_a^b f(y)\, dy\right)^2 \int_a^b \rho(x)\, dx \\
&= \int_a^b \frac{f^2(x)}{\rho(x)}\, dx\ - I^2.
\end{aligned}$$

Ukážeme, že výraz nabývá nejmenší hodnoty pro volbu hustoty ρ_0, která je násobkem absolutní hodnoty funkce f. Taková hustota existuje zřejmě právě jedna, přesněji $\forall\ f \not\equiv 0$ s.j., $|f| \in \mathcal{L}_p([a,b])$ $\exists\,!\ \rho \in \mathcal{L}_p([a,b])$, a lze psát ve tvaru

$$\rho_0(x) = \frac{|f(x)|}{\int_a^b |f(y)|\, dy}.$$

Pro kladnou funkci f je taková volba nasnadě, neboť tím zajistíme konstantnost funkce h. Obecně při volbě této hustoty dostaneme rozptyl

$$\begin{aligned}
\operatorname{var} h(X_1) &= \int_a^b \frac{f^2(x)}{\rho_0(x)}\, dx - I^2 \\
&= \int_a^b \frac{f^2(x)}{|f(x)|} \left(\int_a^b |f(y)|\, dy\right) dx - I^2 \\
&= \left(\int_a^b |f(y)|\, dy\right) \int_a^b \frac{|f(x)|^2}{|f(x)|}\, dx - I^2 \\
&= \left(\int_a^b |f(y)|\, dy\right)^2 - I^2.
\end{aligned}$$

Z Cauchyovy-Schwarzovy nerovnosti potom plyne pro libovolnou hustotu ρ

$$\begin{aligned}
\left(\int_a^b \frac{|f(x)|}{\sqrt{\rho(x)}}\, \sqrt{\rho(x)}\ dx\right)^2 &\le \left(\int_a^b \frac{|f(x)|^2}{\rho(x)}\, dx\right) \left(\int_a^b \rho(x)\, dx\right) \\
\left(\int_a^b |f(x)|\, dx\right)^2 &\le \int_a^b \frac{|f(x)|^2}{\rho(x)}\, dx \\
\left(\int_a^b |f(x)|\, dx\right)^2 - I^2 &\le \int_a^b \frac{f^2(x)}{\rho(x)}\, dx - I^2\ .
\end{aligned}$$

Ukažme, jakým způsobem zapsat metodu v *Mathematice*.

```
Remove[MCM, VarMCM, CIMCM]
Needs["HypothesisTesting`"]

MCM[f_, ρ_, n_Integer] :=
 MCM[f, ρ, {0, 1}, RandomReal[{0, 1}, n]] /; n > 0

MCM[f_, ρ_, {a_, b_}, n_Integer] :=
 MCM[f, ρ, {a, b}, RandomReal[{a, b}, n]] /; n > 0

MCM[f_, ρ_, {a_, b_}, Data_List] :=
 Mean[Map[(f[#] / ρ[#]) &, Data]]

VarMCM[f_, ρ_, {a_, b_}, Data_List] :=
  Variance[Map[(f[#] / ρ[#]) &, Data]]
             Length[Data]

CIMCM[f_, ρ_, {a_, b_}, Data_List, ConfLevel_] :=
 MeanCI[Map[(f[#] / ρ[#]) &, Data], ConfidenceLevel → ConfLevel]
```

Nyní aplikujme metodu na stejný příklad výpočtu integrálu přes [0,1] z funkce

$$f(x) = \frac{e^x - 1}{e - 1} .$$

a volme hustotu $\rho(x) = 2x$ na [0,1]. Analogicky jako v prvním příkladě ponecháme rozsah dat $n = 10\,000$ i jednotlivé výstupy, tzn. nejprve dvojice času potřebného k výpočtu a odhadu integrálu $\{t, \hat{I}_n\}$, dále odhad rozptylu odhadu $\hat{I}_n$, interval spolehlivosti $\{c_l, c_u\}$ pro odhad $\hat{I}_n$ a nakonec vyčíslenou skutečnou hodnotu integrálu I.

```
Remove[f, x]
{a, b} = {0, 1}; f[x_] := (e^x - 1)/(e - 1); ρ[x_] := 2 x
Data = RandomRealFromDensity[ρ, {a, b}, 10 000];

Timing[MCM[f, ρ, {a, b}, Data]]
VarMCM[f, ρ, {a, b}, Data]
CIMCM[f, ρ, {a, b}, Data, 0.95]
∫_a^b f[x] dx // N
```

```
{0.141, 0.417179}

2.71027 × 10^-7

{0.416159, 0.4182}

0.418023
```

Jak je vidět, časové nároky odpovídají předchozí metodě korelovaného výběru, avšak pro dané volby (zde hustoty ρ a funkce g u první metody) měl předchozí odhad přibližně poloviční rozptyl.

4.3 Oblastní výběr

Oblastní výběr (někdy také stratifikovaný) se velice podobá předchozímu postupu. Nejprve si zvolíme oblasti (v jednorozměrném případě intervaly) a ke každé oblasti potom počet simulací, které v dané oblasti provedeme. Tyto simulace odpovídají náhodným výběrům z rovnoměrných rozdělení (tzn. obecné metodě Monte Carlo) na jednotlivých oblastech. Celkový náhodný výběr je zde vlastně nahrazen směsí menších celků, které jsou zastoupeny v předem přesně daném poměru. Kdybychom celkový náhodný výběr konstruovali tak, že nejprve volíme mezi jednotlivými oblastmi náhodně (četnosti nechť odpovídají počtům simulací), dostali bychom předchozí metodu výběru podle důležitosti s hustotou ρ danou jednoduchou (na oblastech konstantní) funkcí. Takto zkonstruovaný náhodný výběr by měl řadu stejných vlastností jako směs používaná při implementaci této metody.

V úloze odhadu integrálu z funkce f přes interval $[a,b]$ tedy nejprve volíme $k \in \mathcal{N}$ a dělení $[a,b]$ dané body $a_0 = a < a_1 < a_2 < \ldots < a_k = b$ a dále volíme pro $j = 1, 2, \ldots, k$ počty simulací n_j na intervalech $[a_{j-1}, a_j]$ tak, aby celkový počet simulací byl $n = n_1 + \ldots + n_k$. Za odhad integrálu potom prohlásíme výraz

$$\hat{I} = \sum_{j=1}^{k} {}_j\hat{I} = \sum_{j=1}^{k} \frac{a_{j-1} - a_j}{n_j} \sum_{i=1}^{n_j} f\left(X_{i,j}\right),$$

kde $X_{i,j}$ jsou náhodné veličiny s rovnoměrným rozdělením $\mathcal{U}([a_{j-1}, a_j])$ pro každé $j = 1, 2, \ldots, k$. Takto sestavený odhad je nestranný, protože

$$\begin{aligned}
\mathrm{E}\,\hat{I} &= \mathrm{E} \sum_{j=1}^{k} \frac{a_{j-1} - a_j}{n_j} \sum_{i=1}^{n_j} f\left(X_{i,j}\right) \\
&= \sum_{j=1}^{k} \frac{a_{j-1} - a_j}{n_j} \sum_{i=1}^{n_j} \mathrm{E}\, f\left(X_{i,j}\right) \\
&= \sum_{j=1}^{k} \left(a_{j-1} - a_j\right) \mathrm{E}\, f\left(X_{1,j}\right) \\
&= \sum_{j=1}^{k} \left(a_{j-1} - a_j\right) \int_R f(x) \frac{1}{a_{j-1} - a_j} \chi_{(a_{j-1} - a_j)}(x)\, \mathrm{d}x \\
&= \sum_{j=1}^{k} \left(a_{j-1} - a_j\right) \int_{a_{j-1}}^{a_j} \frac{1}{a_{j-1} - a_j} f(x)\, \mathrm{d}x \\
&= \sum_{j=1}^{k} \int_{a_{j-1}}^{a_j} f(x)\, \mathrm{d}x = I.
\end{aligned}$$

Rozptyl odhadu jest

$$\begin{aligned}
\mathrm{var}\,\hat{I} &= \mathrm{var} \sum_{j=1}^{k} \sum_{i=1}^{n_j} \frac{a_{j-1} - a_j}{n_j} f\left(X_{i,j}\right) \\
&= \sum_{j=1}^{k} \sum_{i=1}^{n_j} \mathrm{var}\, \frac{a_{j-1} - a_j}{n_j} f\left(X_{i,j}\right) + \sum \mathrm{cov}(\ldots) \\
&= \sum_{j=1}^{k} \frac{\left(a_{j-1} - a_j\right)^2}{n_j^2} \sum_{i=1}^{n_j} \mathrm{var}\, f\left(X_{1,j}\right) \\
&= \sum_{j=1}^{k} \frac{\left(a_{j-1} - a_j\right)^2}{n_j} \left(\mathrm{E} f^2\left(X_{1,j}\right) - \left(\mathrm{E} f\left(X_{1,j}\right)\right)^2\right) \\
&= \sum_{j=1}^{k} \frac{\left(a_{j-1} - a_j\right)^2}{n_j} \left(\int_{a_{j-1}}^{a_j} \frac{1}{a_{j-1} - a_j} f^2(x)\, \mathrm{d}x - \left(\int_{a_{j-1}}^{a_j} \frac{1}{a_{j-1} - a_j} f(x)\, \mathrm{d}x\right)^2\right) \\
&= \sum_{j=1}^{k} \frac{1}{n_j} \left(\left(a_{j-1} - a_j\right) \int_{a_{j-1}}^{a_j} f^2(x)\, \mathrm{d}x - \left(\int_{a_{j-1}}^{a_j} f(x)\, \mathrm{d}x\right)^2\right).
\end{aligned}$$

Označíme-li rozptyly náhodných veličin na jednotlivých oblastech indexovaných $j = 1, 2, \ldots, k$ symbolem

$$ {}_jV^2 = \operatorname{var}\left(a_{j-1} - a_j\right) f\left(X_{1,j}\right), $$

můžeme celkový rozptyl zapsat zjednodušeně jako

$$ \operatorname{var} \hat{I} = \sum_{j=1}^{k} \frac{{}_jV^2}{n_j}. $$

Hodnoty ${}_jV^2$ jakožto rozptyly obyčejných metod Monte Carlo můžeme odhadovat výběrovými rozptyly

$$ {}_jS^2 = \frac{1}{n_j - 1} \sum_{i=1}^{n_j} \left(\left(a_{j-1} - a_j\right) f\left(X_{i,j}\right) - \hat{{}_jI} \right)^2 $$

a celkově tedy aproximovat rozptyl odhadu jako

$$ \begin{aligned} \operatorname{var} \hat{I} &= \sum_{j=1}^{k} \frac{{}_jV^2}{n_j} \\ &\approx \sum_{j=1}^{k} \frac{{}_jS^2}{n_j} \\ &= \sum_{j=1}^{k} \frac{1}{n_j(n_j-1)} \sum_{i=1}^{n_j} \left(\left(a_{j-1} - a_j\right) f\left(X_{i,j}\right) - \hat{{}_jI} \right)^2 \\ &= \sum_{j=1}^{k} \frac{1}{n_j(n_j-1)} \sum_{i=1}^{n_j} \left(\left(a_{j-1} - a_j\right) f\left(X_{i,j}\right) - \frac{a_{j-1} - a_j}{n_j} \sum_{k=1}^{n_j} f\left(X_{k,j}\right) \right)^2 \\ &= \sum_{j=1}^{k} \frac{\left(a_{j-1} - a_j\right)^2}{n_j(n_j-1)} \sum_{i=1}^{n_j} \left(f\left(X_{i,j}\right) - \frac{1}{n_j} \sum_{k=1}^{n_j} f\left(X_{k,j}\right) \right)^2 \end{aligned} $$

Nyní ukažme, jak nejlépe volit rozsahy simulací n_j. Z Cauchyovy-Schwartzovy nerovnosti dostaneme snadno dolní odhad rozptylu, je totiž

$$ \left(\sum_{j=1}^{k} \sqrt{n_j} \, \frac{{}_jV}{\sqrt{n_j}} \right)^2 \le \left(\sum_{j=1}^{k} n_j \right) \left(\sum_{j=1}^{k} \frac{{}_jV^2}{n_j} \right) $$

$$ \left(\sum_{j=1}^{k} {}_jV \right)^2 \le n \operatorname{var} \hat{I} $$

přičemž rovnost nastává právě tehdy, když existuje konstanta c taková, že pro $j = 1,2,\ldots,k$ platí

$$ \frac{{}_jV^2}{n_j} = \frac{1}{c^2} n_j, $$

neboli

$$ {}_jV \; \mathrm{c} = n_j. $$

Hodnotu konstanty c určíme sečtením uvedených podmínek přes j, neboť

$$ \sum_{j=1}^{k} {}_jV \, c = \sum_{j=1}^{k} n_j = n. $$

Pokud již máme dány oblasti a dokážeme odhadnout hodnoty ${}_jV^2$ pro $j = 1,2,\ldots,k$, pak je optimální volit rozsahy výběrů

$$ n_j = {}_jV \; \mathrm{c} = {}_jV \, \frac{n}{\sum_{j=1}^{k} {}_jV}. $$

Jak ale volit jednotlivé oblasti? Nabízí se ekvidistantní dělení. Již takovou stratifikací můžeme dosáhnout podstatné redukce rozptylu, přesto je teoreticky výhodnější volit oblasti tak, aby měly rozptyly odhadů v jednotlivých oblastech přibližně stejnou hodnotu. Uvažujeme-li znovu analogii s předchzí metodou, jde nám o to, co nejlépe se přiblížit jednoduchými funkcemi optimální hustotě ρ pro daný počet intervalů.

Ukažme opět, jakým způsobem zapsat metodu v *Mathematice*. Na vstupu dodáme funkci *f*, stratifikaci intervalu [*a*,*b*] v podobě seznamu $k+1$ hodnot $\{a_0, a_1, \ldots, a_k\}$, kde $a_0 = a < a_1 < a_2 < \ldots < a_k$, a dále buďto seznam $\{n_1, n_2, \ldots, n_k\}$ počtu simulací, anebo rovnou seznam generovaných dat $\{Data_1, Data_2, \ldots, Data_k\}$ pro jednotlivé intervaly.

```
Remove[MCM, VarMCM]

MCM[f_, Str_List, nList_?VectorQ] :=
 MCM[f, Str, MapThread[RandomReal[{#1, #2}, #3] &,
   {Most[Str], Rest[Str], nList}]]

MCM[f_, Str_List, Data_List] :=
 Total[MapThread[(#2 - #1) Mean[#3] &,
   {Most[Str], Rest[Str], Map[f, Data, {2}]} ]]

VarMCM[f_, Str_List, Data_List] :=
 Total[MapThread[Variance[(#2 - #1) #3]/Length[#3] &,
   {Most[Str], Rest[Str], Map[f, Data, {2}]}]]
```

Pro srovnání s předchozími metodami odhadujme opět integrál přes [0,1] z funkce

$$f(x) = \frac{e^x - 1}{e - 1}.$$

Volme $k = 4$ a stratifikaci danou seznamem bodů {0, 0.36, 0.62, 0.83, 1}. V každé oblasti vybírejme náhodně stejný počet 2500 bodů, takže celkový rozsah generovaných dat zůstane $n = 10\,000$. Ke vstupním datům ještě poznamenejme, že seznam rozsahů dat *nList* definujeme pro pohodlnější editaci tak, že stačí zadat pouze celkový požadovaný počet dat *n* a poměry počtu simulací (první argument funkce *Normalize*). Výstupy budou analogické jako v předchozích podkapitolách.

```
Remove[f, x]

Str = {0, 0.36, 0.62, 0.83, 1};
f[x_] := (E^x - 1)/(E - 1)

nList = 10 000 Normalize[{1, 1, 1, 1}, Norm[#, 1] &];
Data = MapThread[RandomReal[{#1, #2}, #3] &,
   {Most[Str], Rest[Str], nList}];
```

```
Timing[MCM[f, Str, Data]]
VarMCM[f, Str, Data]
∫_First[Str]^Last[Str] f[x] dx // N
```

```
{0.094, 0.41828}
```

$$5.64268 \times 10^{-7}$$

```
0.418023
```

Na závěr podkapitoly poznamenejme, že volbou stratifikace $\{a,b\}$ dostaneme výsledky ekvivalentní obyčejné metodě Monte Carlo a funkce *MCM* zde proto zobecňuje funkci se stejným názvem deklarovanou dříve. Zdůrazněme ale, že parametr *Data* ve zdejší deklaraci musí mít tvar matice. Co se týče srovnání výsledků s obyčejnou metodou MC, pro zvolenou stratifikaci (a daný příklad), pracují obě metody přibližně stejnou dobu, ale při stratifikaci dosáhneme zhruba 15-krát menšího rozptylu.

■ 4.4 Antitetické proměnné

Uvažujme, že máme dva nestranné odhady $\hat{I}_1$ a $\hat{I}_2$ odhadující stejnou hodnotu. Jestliže jsou záporně korelovány, pak odhadem

$$\hat{I} = \tfrac{1}{2}\left(\hat{I}_1 + \hat{I}_2\right)$$

se zachová střední hodnota

$$\mathrm{E}\,\hat{I} = \tfrac{1}{2}\left(\mathrm{E}\,\hat{I}_1 + \mathrm{E}\,\hat{I}_2\right) = \mathrm{E}\,\hat{I}_1 = \mathrm{E}\,\hat{I}_2$$

a pro rozptyl platí

$$\begin{aligned}\mathrm{var}\,\hat{I} &= \tfrac{1}{4}\,\mathrm{var}\,\hat{I}_1 + \tfrac{1}{2}\,\mathrm{cov}\left(\hat{I}_1, \hat{I}_2\right) + \tfrac{1}{4}\,\mathrm{var}\,\hat{I}_2 \\ &\le \tfrac{1}{2}\,\max\left(\mathrm{var}\,\hat{I}_1, \mathrm{var}\,\hat{I}_2\right).\end{aligned}$$

Toho využívá metoda antitetických proměnných tak, že k náhodným datům pocházejících z nějakého rozdělení se dopočítávají antitetické protějšky, které zastupují data ze symetrického rozdělení (symetrického podle střední hodnoty). Shoduje-li se symetrické rozdělení s původním, pak náhodná data spolu s jejich antitetickými protějšky nahrazují realizaci náhodného výběru s dvojnásobným rozsahem a teoretická hodnota rozptylu odhadu počítaného na základě těchto dat nepřesáhne polovinu hodnoty rozptylu původního odhadu. U normálního rozdělení jsou antitetické páry symetrické podle nuly (nulového vektoru), proto například u Wienerova procesu považujeme za symetrický protějšek takový proces, který má stejné přírůstky, ale s opačným znaménkem.

Náhodná veličina, která vznikne sečtením symetrického páru, má zřejmě nulový rozptyl. Kdybychom na rozdělení, ze kterého pár pochází, aplikovali lineární transformaci, dostali bychom sečtením taktéž veličinu s nulovým rozptylem. Metoda antitetických proměnných proto přináší nejlepší výsledky při použití na funkce f blízké lineárním (zde máme na mysli i funkce lineární po odečtení konstanty). Obecněji spočívá podstata redukce rozptylu uvedeným postupem v následujícím. Nechť náhodná veličina z níž generujeme data má rozdělení X se střední hodnotou $\mu = \mathrm{E}\,X$ a nechť veličiny $X - \mu$ a $\mu - X$ jsou stejně rozdělené. Rozepišme si funkci f jako součet její symetrické a antisymetrické části

$$f(x) = f_s(x) + f_a(x),$$

kde f_s a f_a splňují

$$f_s(x-\mu) = \frac{f(x-\mu)+f(\mu-x)}{2},\ f_a(x-\mu) = \frac{f(x-\mu)-f(\mu-x)}{2}.$$

Potom veličiny $f_s(X)$ a $f_a(X)$ jsou nekorelované, neboť

$$\begin{aligned}\operatorname{cov}(f_s(X), f_a(X)) &= \mathrm{E}\, f_s(X)\, f_a(X) - \mathrm{E}\, f_s(X)\, \mathrm{E}\, f_a(X)\\ &= \mathrm{E}\, \frac{f^2(X) - f^2(2\mu - X)}{4} - \mathrm{E}\, \frac{f(X)+f(2\mu-X)}{2}\, \mathrm{E}\, \frac{f(X)-f(2\mu-X)}{2}\\ &= \frac{\mathrm{E}\, f^2(X) - \mathrm{E}\, f^2(X)}{4} - \frac{\mathrm{E}\, f(X) + \mathrm{E}\, f(X)}{2}\, \frac{\mathrm{E}\, f(X) - \mathrm{E}\, f(X)}{2} = 0,\end{aligned}$$

a proto můžeme psát i rozptyl $f(X)$ jako součet

$$\operatorname{var} f(X) = \operatorname{var} f_s(X) + \operatorname{var} f_a(X).$$

Jenomže pro ryze antisymetrickou funkci aplikovanou na antitetický pár platí

$$f_a(X) + f_a(2\mu - X) = \frac{f(X) - f(2\mu - X)}{2} + \frac{f(2\mu - X) - f(2\mu - (2\mu - X))}{2} = 0,$$

nezávisle na hodnotách X. Rozptyl odhadu integrálu pro n párů tak můžeme zapsat jako

$$\begin{aligned}\operatorname{var} \hat{I}_n &= \operatorname{var} \frac{b-a}{n} \sum_{i=1}^{n} \frac{f(X_i)+f(2\mu - X_i)}{2}\\ &= \frac{(b-a)^2}{(2n)^2} \operatorname{var} \sum_{i=1}^{n} (f(X_i) + f(2\mu - X_i))\\ &= \frac{(b-a)^2}{4n^2} \left(\sum_{i=1}^{n} (f(X_i) + f(2\mu - X_i)) + \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} \operatorname{cov}(\ldots) \right)\\ &= \frac{(b-a)^2}{4n} \operatorname{var} (f(X_1) + f(2\mu - X_1))\\ &= \frac{(b-a)^2}{4n} \operatorname{var} (f_s(X_1) + f_a(X_1) + f_s(2\mu - X_1) + f_a(2\mu - X_1))\\ &= \frac{(b-a)^2}{4n} \operatorname{var} (f_s(X_1) + f_s(2\mu - X_1)).\end{aligned}$$

Tím jsme ukázali, že metoda poskytne obecně menší rozptyl pro funkce se symetrickou částí blízké konstantní funkci. Množina funkcí lineárních až na konstantu tvoří jen malou podmnožinu třídy funkcí, z nichž lze touto metodou odhadnout integrál přesně.

Nejelementárnější použití metody pro odhad integrálu z funkce na intervalu můžeme sice implementovat tak, že nejprve vytvoříme antitetická data, která následně zpracujeme obyčejnou metodou MC, přikloníme se však k následující paměťově úspornější variantě.

```
Remove[MCM, VarMCM]

MCM[f_, n_Integer] := MCM[f, {0, 1}, RandomReal[{0, 1}, n]] /; n > 0

MCM[f_, {a_, b_}, n_Integer] :=
 MCM[f, {a, b}, RandomReal[{a, b}, n]] /; n > 0

MCM[f_, {a_, b_}, Data_List] :=
 (b - a)/2 Mean[Map[f[#] + f[a + b - #] &, Data]]
```

```
VarMCM[f_, {a_, b_}, Data_List] :=
 ((b - a)/2)^2 Variance[Map[f[#] + f[a + b - #] &, Data]]/Length[Data]
```

Nyní opět srovnejme metodu na příkladu odhadu integrálu přes [0,1] z funkce

$$f(x) = \frac{e^x - 1}{e - 1}.$$

Ponechme rozsah vstupních dat $n = 10\,000$ i formu jednotlivých výstupů.

```
Remove[f, x]
{a, b} = {0, 1}; f[x_] := (e^x - 1)/(e - 1)
Data = RandomReal[{a, b}, 10 000];

Timing[MCM[f, {a, b}, Data]]
VarMCM[f, {a, b}, Data]
∫_a^b f[x] dx // N
```

```
{0.219, 0.417792}
1.31311 × 10^-7
0.418023
```

Výsledky ukazují, že ve srovnání s obyčejnou metodou MC je tato pro stejné rozsahy dat výrazně pomalejší, rozptyl se však snížil více než 60-krát.

■ 4.5 Regresní metody

Regresní metody umožňují snadno kombinovat předešlé postupy. Pro začátek připomeňme některé poznatky z regresní analýzy, které dále využijeme. Základy regresní analýzy lze nastudovat ze skript [D05]. Budeme se zabývat modelem lineární regrese

$$\Upsilon = A\,\beta + \epsilon\,,$$

kde pro $k, m \in \mathcal{N}$ je $\Upsilon = \{Y_1, \ldots, Y_k\}^T$ vektor pozorování, A je daná matice typu $k \times m$ charakterizující použitý regresní model, $\beta = \{\beta_1, \ldots, \beta_m\}^T$ je vektor neznámých hodnot (množina parametrů, které chceme odhadnout), a $\epsilon = \{\epsilon_1, \ldots, \epsilon_k\}^T$ je vektor chyb s nulovou střední hodnotou. Jeho kovarianční matici označme $var\,\epsilon = var\,\Upsilon = V$ a předpokládejme dále, že V je pozitivně definitní. Upozorněme, že stejně jako vektor β je matice A deterministická, tedy konstantní vzhledem k pravděpodobnosti v uvažovaném stochastickém modelu, naprotitomu Υ a ϵ jsou náhodné vektory. Předpokládejme ještě, že $k \geq m$ a že hodnost matice A je m. Potom

$$\hat{\beta} = \left(A^T\,V^{-1}\,A\right)^{-1}\left(A^T\,V^{-1}\,\Upsilon\right)$$

je nejlepším nestranným odhadem β s kovarianční maticí

$$\operatorname{var}\hat{\beta} = \left(A^T\, V^{-1}\, A\right)^{-1}.$$

Při praktickém použití často nahrazujeme matici V jejím odhadem. V takovém případě už odhad β obecně není nestranný, přesto má dobré asymptotické vlastnosti. V příkladu, který bude následovat, využijeme s výhodou toho, že k výpočtu výběrového průměru

$$\overline{\Upsilon} = \frac{1}{n} \sum_{i=1}^{n} \Upsilon_i$$

z náhodného výběru $\{\Upsilon_1, \ldots, \Upsilon_n\}$ nepotřebujeme simulovat $n \times k$, ale pouze n hodnot. Použijeme totiž stejné realizace výběru pro každou složku vektoru, protože simulace v jednotlivých složkách spolu nesouvisí. Bez větších výpočetních nároků tak odhadneme kovarianční matici V výrazem

$$\hat{V} = \frac{1}{n-1} \sum_{i=1}^{n} \left(\Upsilon_i - \overline{\Upsilon}\right)\left(\Upsilon_i - \overline{\Upsilon}\right)^T.$$

Místo původního odhadu β pak můžeme na základě pozorování $\{\Upsilon_1, \ldots, \Upsilon_n\}$ dostat přesnější

$$\hat{\beta} = \left(A^T\, \hat{V}^{-1}\, A\right)^{-1} \left(A^T\, \hat{V}^{-1}\, \overline{\Upsilon}\right),$$

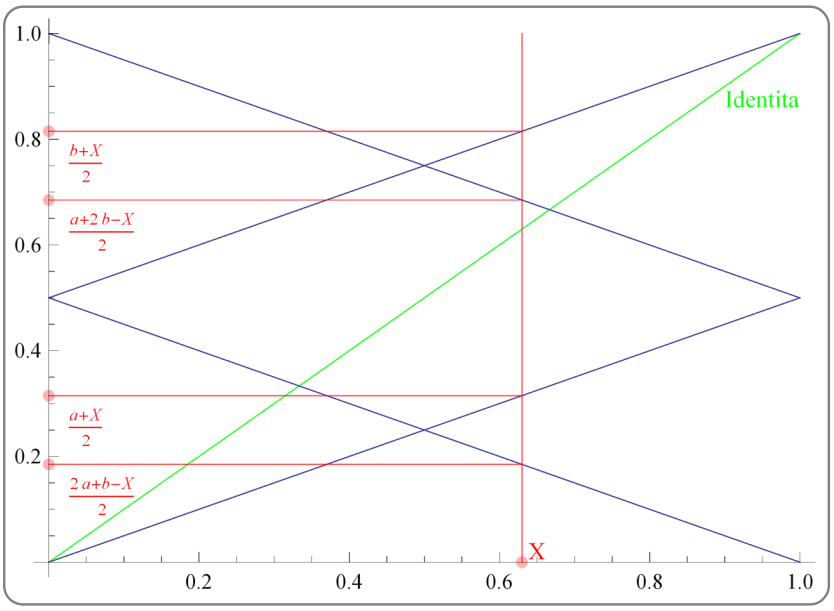
který má asymtoticky kovarianční matici

$$\operatorname{var}\hat{\beta} = \frac{1}{n}\left(A^T\, \hat{V}^{-1}\, A\right)^{-1}.$$

Nyní ukažme, jak aplikovat popsaný regresní model v úloze výpočtu jednorozměrného integrálu. V modelu volme $m = 1$, $k = 2$, A jako matici jednotek a nechť

$$Y_1 = \frac{(b-a)}{2}\left(f(X) + f(a+b-X)\right),$$
$$Y_2 = \frac{(b-a)}{4}\left(f\left(\frac{a+X}{2}\right) + f\left(\frac{a+2\,b-X}{2}\right) + f\left(\frac{b+X}{2}\right) + f\left(\frac{2\,a+b-X}{2}\right)\right),$$

kde náhodná veličina X má rovnoměrné rozdělení $\mathcal{U}([a,b])$. Pro lepší představu o argumentech funkce přikládám obrázek.



Obr. 4.1. Antitetické transformace veličiny X.

Budeme odhadovat jediný parametr

$$\beta_1 = I = \int_a^b f(x)\, dx.$$

Symbolicky můžeme psát

$$\begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \beta_1 + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \end{pmatrix},$$

odkud ihned vidíme, že chyby

$$\epsilon_1 = Y_1 - \beta_1 = \tfrac{(b-a)}{2}\,(f(X) + f(a+b-X)\,) - \int_a^b f(x)\, dx\,,$$
$$\epsilon_2 = Y_2 - \beta_1 = \tfrac{(b-a)}{4}\left(f\left(\tfrac{a+X}{2}\right) + f\left(\tfrac{a+2\,b-X}{2}\right) + f\left(\tfrac{b+X}{2}\right) + f\left(\tfrac{2\,a+b-X}{2}\right)\right) - \int_a^b f(x)\, dx\,,$$

mají požadované vlastnosti, protože Y_1 i Y_2 jsou nestrannými odhady integrálu I. Před vlastní implementací dodejme, že pro odhad $\hat{V}$ voláme v *Mathematice* jednoduše vestavěnou funkci *Covariance* na seznam n simulací k-rozměrných náhodných vektorů $\{\Upsilon_1, \ldots, \Upsilon_n\}^T$, jejím vstupem tak bude matice s k řádky a n sloupci. Zavedeme nejprve obecné funkce (pro případ $m = 1$, obecné k, s A maticí jednotek), které mají na vstupu pouze data (generovaná z $\mathcal{U}([a,b])$) a k-rozměrný vektor odhadů integrálu Υ. Budeme předpokládat, že vstupní informace o funkci f budou obsaženy právě v tomto vektoru odhadů.

```
Remove[MCM, VarMCM]

MCM[Est_, Data_List] := Module[
  {Y = Map[Est, Data], InvV},
  InvV = Inverse[Covariance[Y]];
  Total[InvV, 2]^-1 (Total[InvV].Mean[Y]) ]

VarMCM[Est_, Data_List] := Module[
  {Y = Map[Est, Data], InvV},
  InvV = Inverse[Covariance[Y]];
  (Length[Data] Total[InvV, 2])^-1]
```

Uvedené deklarace funkcí předpokládají, že vektor odhadů Υ reprezentujeme ryzí funkcí, aby bylo možné na něj snadno mapovat generovaná data. Zároveň chceme, aby Υ respektoval parametry f a $[a,b]$, takže model s odhady Y_1 a Y_2 můžeme promítnout do implementace následovně.

```
Remove[Est]

Est[f_, {a_, b_}] := {
    (b - a)/2 (f[#] + f[a + b - #]),
    (b - a)/4 (f[(a + #)/2] + f[(a + 2 b - #)/2] + f[(b + #)/2] + f[(2 a + b - #)/2])
  } &
```

Nyní se podíváme, jaké výsledky obdržíme při aplikaci metody v několikrát uvedeném příkladě s funkcí f na intervalu [0,1] danou předpisem

$$f(x) = \frac{e^x - 1}{e - 1}.$$

Pro srovnání s předchozími metodami zachováme rozsah dat $n = 10\,000$. Na výstupu obdržíme znovu dvojici $\{t, \hat{I}\}$, dále rozptyl odhadu (přesněji jen odhad teoretické hodnoty) a nakonec skutečnou hodnotu integrálu *I*.

```
Remove[f, x]

{a, b} = {0, 1}; f[x_] := (e^x - 1)/(e - 1)
Data = RandomReal[{a, b}, 10 000];

Timing[MCM[Est[f, {a, b}], Data]]
VarMCM[Est[f, {a, b}], Data]
∫_a^b f[x] dx // N
{0.859, 0.418023}
2.42378 × 10^-13
0.418023
```

I když tato metoda pracuje zhruba 10-krát pomaleji než obyčejná metoda MC, poměr rozptylů 34×10^6 jednoznačně převáží eficienci ve prospěch regresní metody. Řádově jsme vlastně v daném příkladě zlepšili přesnost nejméně o 3 desetinná místa.

5 Metody quasi Monte Carlo

Metoda *quasi Monte Carlo* (QMC) se od klasické metody v zásadě liší pouze tvorbou náhodných dat. U metod QMC už snad ani nemůže být řeč o náhodnosti, posloupnost dat totiž vytváříme naprosto bez ohledu na vnitřní korelovanost a předem vybrané podposloupnosti nemusí nutně respektovat požadované rozdělení. Na data pohlížíme jako na celek s předem daným rozsahem, podobně jako jsme s nimi doposud pracovali, a generujeme je za jediným účelem minimalizace chyby odhadu integrálu na co nejširší třídě funkcí.

Protože u metod QMC se generované body vybírají velice pečlivě, v teorii se omezíme na odhad integrálu z funkce $f\colon [0,1]^d \to \mathcal{R}$ přes její doménu, $d \in \mathcal{N}$. Integrál odhadujeme z dat analogicky jako u klasické metody MC výběrovým průměrem z hodnot funkce f mapované na data.
Data z $[0,1]^d$ generujeme sofistikovanými metodami, které vrací různé sekvence s ohledem na předem daný rozsah n a především dimenzi d tak, aby měla co nejnižší *diskrepanci*. Tento požadavek formálně odpovídá výběru takových bodů, které vyplní *hyperkrychli* $[0,1]^d$ rovnoměrně. Zdůrazněme, že není zdaleka optimální používat metodu zvlášť na jednotlivé složky ani podprostory, které skládáme zpět direktním součtem.

Význam metody QMC spočívá především v možnosti shora teoreticky odhadnout chybu vzniklou integrací, narozdíl od klasických metod, kde se můžeme opírat pouze o pravděpodobnostní rozdělení chyby, která obecně nemá konečné meze. Dále uvedeme něco z teorie z práce Glassermana [G04].

5.1 Diskrepance

Nechť $\mathcal{A}$ je systém λ-měřitelných podmnožin $[0,1]^d$. Potom *diskrepanci* množiny bodů $\{x_1, \dots, x_n\}$ vzhledem k $\mathcal{A}$ definujeme jako

$$D(\{x_1, \dots, x_n\}, \mathcal{A}) = \sup_{A \in \mathcal{A}} \left| \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{(x_i \in A)} - \lambda(A) \right|.$$

Slovy bychom řekli, že diskrepance je suprémum přes chyby odhadů integrálů (na hyperkrychli $[0,1]^d$) z charakteristických funkcí množin z $\mathcal{A}$, kde odhadujeme pro každé A metodou MC z dat $\{x_1, \dots, x_n\}$, tj. výběrovým průměrem z hodnot χ_A mapovaných na data. Volíme-li za $\mathcal{A}$ všechny elementární intervaly (též d-rozměrné obdélníky, tj. všechny kartézské součiny d intervalů), mluvíme o *běžné diskrepanci* $D(x_1, \dots, x_n)$, nebo také *extremální*. Pokud v levých krajních bodech volíme vždy 0 (obdélníky mají vždy jeden vrchol v počátku 0_d), dostaneme *diskrepanci s hvězdičkou* $D^*(x_1, \dots, x_n)$. *Izotropní diskrepance* například pracuje se všemi konvexními podmnožinami $[0,1]^d$. Lze ukázat, že platí

$$D^*(x_1, \dots, x_n) \le D(x_1, \dots, x_n) \le 2^d D^*(x_1, \dots, x_n),$$

dále uveďme jen pro ilustraci některé další výsledky převzaté z [G04].

Například pro dimenzi $d = 1$ lze diskrepance pro body $\{x_1, \ldots, x_n\}$ odhadnout zdola

$$D^*(x_1, \ldots, x_n) \geq \frac{1}{2n}, \quad D(x_1, \ldots, x_n) \geq \frac{1}{n},$$

přičemž pro oba případy lze minima dosáhnout volbou bodů

$$x_i = \frac{2i-1}{2n}, \quad i = 1, 2, \ldots, n.$$

Výsledná integrace pak odpovídá *lichoběžníkovému pravidlu* pro numerické integrování. Zdůrazněme, že sekvence bodů se zde vytváří na základě předem známého rozsahu bodů n a pro různá n obdržíme obecně různé sekvence. Výraz také neříká, jak volit nekonečnou sekvenci bodů. Předpokládejme naopak, že budeme mít dánu nekonečnou sekvenci bodů a zajímá nás diskrepance prvních n prvků. Jak uvádí [G04], potom platí

$$D(x_1, \ldots, x_n) \geq D^*(x_1, \ldots, x_n) \geq \frac{c \log n}{n}$$

pro nekonečně mnoho n, kde konstanta c nezávisí na n.

■ 5.2 Odhad chyby při integrování metodou QMC

Nechť f je konečná funkce na $[0,1]^d$ a J libovolný d-rozměrný obdélník v $[0,1]^d$, tj. množina bodů

$$J = \left[{}^{-1}u_1, {}^{+1}u_1\right] \times \left[{}^{-1}u_2, {}^{+1}u_2\right] \times \ldots \times \left[{}^{-1}u_d, {}^{+1}u_d\right],$$

kde $0 \leq {}^{-1}u_j \leq {}^{+1}u_j \leq 1$, pro $j = 1, \ldots, d$. Každý z vrcholů obdélníku J udává vektor ve tvaru $\mathfrak{u} = \{{}^{s_1}u_1, {}^{s_2}u_2, \ldots, {}^{s_d}u_d\}$, kde $s_j \in \{-1, 1\}$ pro $j = 1, \ldots, d$. Označíme-li V množinu všech 2^d vrcholů J, můžeme definovat přírůstek f přes J jako

$$\Delta_J(f) = \sum_{\mathfrak{u} \in V} \left(\prod_{j=1}^{d} s_j\right) f(\mathfrak{u}).$$

Dále označme $\mathcal{P} = \{J_i\}_{i \in I}$ množinu obdélníku ve tvaru J, které tvoří stratifikaci (dělení) hyperkrychle $[0,1]^d$, což lze charakterizovat například tak, že systém $\mathcal{P}' = \{J'_i\}_{i \in I}$ s prvky

$$J'_i = \left[{}^{-1}{}_i u_1, {}^{+1}{}_i u_1\right) \times \left[{}^{-1}{}_i u_2, {}^{+1}{}_i u_2\right) \times \ldots \times \left[{}^{-1}{}_i u_d, {}^{+1}{}_i u_d\right),$$

kde $0 \leq {}^{-1}{}_i u_j < {}^{+1}{}_i u_j < 1$, pro $j = 1, \ldots, d$, $i \in I$ tvoří disjunktní pokrytí $[0, 1)^d$, a stratifikaci $\mathcal{P}$ pak dostaneme aplikací uzávěru na každý prvek $\mathcal{P}'$, tedy $J_i = \overline{J'_i}$ pro každé $i \in I$. Nyní můžeme zavést míru variace funkce f jako suprémum přes všechny možné stratifikace $\mathcal{P}$ hyperkrychle $[0, 1]^d$

$$V^{(d)}(f) = \sup_{\mathcal{P}} \sum_{J \in \mathcal{P}} |\Delta_J(f)|.$$

Jestliže f má na $[0,1]^d$ spojité parciální derivace, pak

$$V^{(d)}(f) = \int_0^1 \ldots \int_0^1 \left| \frac{\partial^d f}{\partial x_1 \ldots \partial x_d} \right| d x_1 \ldots d x_d,$$

nicméně $V^{(d)}(f)$ je dobře definována nejen pro diferencovatelné, ale pro všechny konečné funkce f na $[0,1]^d$. Konečně pro libovolné $k \in \{1, 2, \dots, d\}$ a k libovolně vybraných souřadnic splňujících $1 \le i_1 < \dots < i_k \le d$ označme $f|_{\mathcal{S}(i_1, \dots, i_k)}$ restrikci funkce f na množinu bodů $\mathbf{u} = \{u_1, u_2, \dots, u_d\}$, kde $u_j \in [0,1]$ pro $j \in \{i_1, \dots, i_k\}$ a $u_j = 1$ jinak, a definujme *Hardyovu-Krauseovu variaci* funkce f předpisem

$$V(f) = \sum_{k=1}^{d} \sum_{1 \le i_1 < \dots < i_k \le d} V^{(k)}\left(f|_{\mathcal{S}(i_1, \dots, i_k)}\right).$$

Nyní již můžeme uvést výraz pro stanovení horní meze chyby při integrování metodami QMC známý jako *Koksmova-Hlawkova mez*. Při zavedeném značení platí

$$\left| \frac{1}{n} \sum_{i=1}^{n} f(x_i) - \int_{[0,1]^d} f(\mathbf{y})\, d\mathbf{y} \right| \le V(f)\, D^*(x_1, \dots, x_n).$$

Při praktických výpočtech je ale obtížné stanovit hodnoty $V(f)$ i $D^*(x_1, \dots, x_n)$ v porovnání například s výpočtem odhadu samotného integrálu nebo stanovením intervalu spolehlivosti. Navíc Koksmova-Hlawkova mez zpravidla hrubě nadhodnocuje skutečnou chybu při odhadování.

5.3 Posloupnosti s nízkou diskrepancí

Za *posloupnost bodů s nízkou diskrepancí* považujeme posloupnost $\{x_n\}_{n=1}^{\infty}$, $x_i \in \mathcal{R}^d$ pro $i \in \mathcal{N}$, která splňuje pro libovolné $N \in \mathcal{N}$

$$D^*(x_1, \dots, x_N) \le c_d \frac{(\log N)^d}{N},$$

kde konstanta c_d nezávisí na N. Data $\{x_1, \dots, x_N\}$ z této posloupnosti někdy označujeme jako *quasi náhodná data*.

Uveďme jednoduchý příklad posloupnosti s nízkou diskrepancí známou jako *Haltonova sekvence* pracující výhodně se zápisy čísel v soustavách se základem b. Nechť $b \in \mathcal{N}$, $b \ge 2$, je báze a $\{0, 1, \dots, b-1\}$ množina číslic, se kterými pracujeme v soustavě dané zvolenou bází. Hodnotu každého čísla lze zapsat jako řadu

$$q = \sum_{k=-\infty}^{\infty} {}^{k}d_b(q)\; b^k,$$

kde ${}^{k}d_b(q)$ označuje číslici na k-té pozici (v řádu b^k) v "desetinném" zápise čísla q v soustavě s bází b. Každé racionální číslo $q \in \mathcal{Q}$ lze navíc zapsat jednoznačně jako konečný součet, proto pro $q \in \mathcal{Q}$ můžeme definovat funkci

$$\phi_b(q) = \sum_{k=-\infty}^{\infty} {}^{k}d_b(q)\; b^{-k-1}.$$

Slovy bychom řekli, že se jedná o záměnu číslic v "desetinném" (pro $q \in \mathcal{Q}$ navíc konečném) zápise nějakého čísla při bázi b symetricky podle "desetinné tečky". Pro nás je v tuto chvíli podstatné, že ϕ_b zobrazuje množinu přirozených čísel $\mathcal{N}$ do intervalu $[0,1]$ a navíc tak, že posloupnost $\{\phi_b(n)\}_{n=1}^{\infty}$ má nízkou diskrepanci. Podívejme se na několik prvních členů posloupnosti $\{\phi_2(n)\}_{n=1}^{\infty}$, známé jako *van der Corputova posloupnost*.

```
ϕ[base_, n_Integer] :=
 FromDigits[{Reverse[IntegerDigits[n, base]], 0}, base]

TableForm[Map[{#, BaseForm[#, 2],
     BaseForm[ϕ[2, #] // N, 2], ϕ[2, #]} &,
   Range[7]] // Transpose, TableDepth → 2, TableHeadings → {
   {"n", "(n)₂", "(ϕ₂(n))₂", "ϕ₂(n)"}, None}]
```

n	1	2	3	4	5	6	7
$(n)_2$	1_2	10_2	11_2	100_2	101_2	110_2	111_2
$(\phi_2(n))_2$	0.1_2	0.01_2	0.11_2	0.001_2	0.101_2	0.011_2	0.111_2
$\phi_2(n)$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{5}{8}$	$\frac{3}{8}$	$\frac{7}{8}$

Chceme-li podobným způsobem vyplnit rovnoměrně hyperkrychli $[0,1]^d$, stačí volit d navzájem nesoudělných čísel $p_1, \ldots, p_d$ a definovat *Haltonovu sekvenci* bodů

$$x_i = \{\phi_{p_1}(i), \ \ldots \ , \phi_{p_d}(i)\}, \quad i = 1, 2, \ldots$$

Protože zřejmě pro vyšší dimenze preferujeme co nejnižší základy, volíme obvykle $p_1, \ldots, p_d$ jako prvních d prvočísel. Všimněme si, že indexováním posloupnosti od $i = 0$ lze volitelně zahrnout do sekvence i počátek. *Hammersleyho množinu bodů* v $[0,1]^d$ s předem daným rozsahem n lze definovat rozšířením prvních n členů Haltonovy sekvence pro dimenzi d–1 jako

$$y_i = \left\{\frac{i}{n}, \phi_{p_1}(i), \ \ldots \ , \phi_{p_{d-1}}(i)\right\}, \quad i = 0, 1, \ldots, n-1,$$

a drobnou úpravou *Woźniakowského množinu bodů* nebo také *posunutou Hammersleyho množinu bodů*

$$z_i = \left\{\frac{i-1+\eta}{n}, \phi_{p_1}(i), \ \ldots \ , \phi_{p_{d-1}}(i)\right\}, \quad i = 1, \ldots, n \ ,$$

která navíc obsahuje parametr $\eta \in \mathcal{R}$ omezený podmínkou $0 \le i - 1 + \eta < 1$. Obecněji se zavádí *Woźniakowského množina* s m-periodickou Haltonovou posloupností (viz. [U03]), která má se standartní Haltonovou posloupností společných prvních m členů a dále se dodefinuje jako

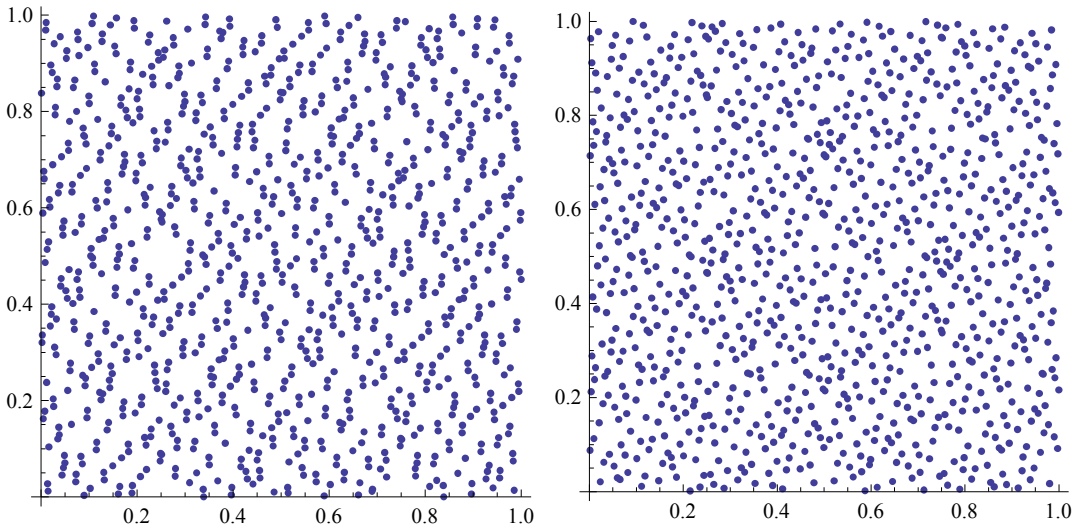
$$x_{i+m} = x_i, \quad i = 1, 2, \ldots$$

Další posloupnosti s nízkou diskrepancí jsou například *Faureovy*, *Sobolovy* nebo *Niederreiterovy* sekvence. Jak je uvedeno v [S02] , všechny splňují

$$N.D^*(x_1, \ldots, x_n) \le c_d(\log N)^d + O\left((\log N)^{d-1}\right).$$

Pro dimenzi $d = 2$ lze například u Haltonovy sekvence pracovat s konstantou $c_2 = 0.2602$.
Program *Mathematica* nabízí možnost generování *Niederreiterových* a *Sobolových* sekvencí jako jedno z možných nastavení funkce *RandomReal*. Protože ale při generování pracuje s Intelovskými knihovnami *MKL*, omezuje tak jejich použití na vybrané platformy. Následuje ukázka obou sekvencí (Niederrieterovy vlevo, Sobolovy vpravo) s rozsahem $N = 1000$ pro dimenzi $d = 2$. Na první pohled rozpoznáme v obou ukázkách jistou pravidelnost či vzor.

```
{Map[ListPlot[BlockRandom[
       SeedRandom[Method → {"MKL", Method → {
             #, "Dimension" → 2}}];
       RandomReal[{0, 1}, {1000, 2}]],
      AspectRatio → 1] &
    , {"Niederreiter", "Sobol"}]
 } // TableForm
```



Pro srovnání definujme dále funkci *MCM* odhadující integrál z *d*-rozměrné funkce na $[0,1]^d$ klasickou metodou MC a analogickou funkci *QMC* využívající quasi náhodná čísla některého z generátorů výše.

```
Remove[MCM, QMC]
Needs["HypothesisTesting`"]

Options[MCM] = {ConfidenceLevel → 95 / 100};

MCM[f_, d_, n_Integer, o___?OptionQ] := MCM[f,
   RandomReal[1, {n, d}], o] /; n > 0

MCM[f_, Data_List, o___?OptionQ] := Module[
  {distEstim = MapThread[f, Transpose[Data]],
   α = ConfidenceLevel /. Flatten[{o, Options[MCM]}]},
  TableForm[{{{Mean[distEstim],
      Variance[distEstim] / Length[Data],
      ToString[MeanCI[distEstim, ConfidenceLevel → α]]
     }}, TableHeadings → {None,
     {"Estimation", "Variance",
      ToString[100 α] <> "% Confidence Interval"}}]]
```

```
Options[QMC] = {
    ConfidenceLevel → 95 / 100,
    Method → "Sobol"};

QMC[f_, d_, n_Integer, o___?OptionQ] := QMC[f,
  BlockRandom[
   SeedRandom[Method → {"MKL", Method → {
         Method /. Flatten[{o, Options[QMC]}],
          "Dimension" → d}}];
    RandomReal[1, {n, d}]], o]

QMC[f_, Data_List, o___?OptionQ] := MCM[f, Data, o]
```

Srovnávejme na příkladu s funkcí f: $[0,1]^3 \to \mathcal{R}$ a rozsahem dat $n = 10\,000$.

```
Remove[f, x, y, z]

d = 3;
f[x_, y_, z_] := (ℯ^x - 1)/(ℯ - 1) Sqrt[y] - z^2 + 1
n = 10 000;

Timing[MCM[f, d, n]]
Timing[QMC[f, d, n, Method → "Sobol"]]
Timing[QMC[f, d, n, Method → "Niederreiter"]]
∫_0^1 (∫_0^1 (∫_0^1 f[x, y, z] dx) dy) dz // N
```

```
{0.188, Estimation  Variance      95% Confidence Interval
        0.937963    0.0000137934  {0.930682, 0.945243}}

{0.187, Estimation  Variance      95% Confidence Interval
        0.945382    0.0000139655  {0.938057, 0.952707}}

{0.188, Estimation  Variance      95% Confidence Interval
        0.945263    0.0000139574  {0.93794, 0.952586}}

0.945349
```

Ve výsledcí prakticky nelze vypozorovat žádné větší odlišnosti. Nicméně kdybychom stejné výpočty provedli vícekrát, všimli bychom si, že hodnoty odhadu rozptylu metody QMC nejsou tolik rozptýlené jako u klasické metody MC.

6 Aplikace metody Monte Carlo

Nejprve se naučíme základní postupy numerického integrování v programu *Mathematica* a dále se zaměříme na nejjednodušší příklady použití metod Monte Carlo ve financích.

■ 6.1 Numerické integrování v Mathematice

V souvislosti s numerickým počítáním integrálů v *Mathematice* nemůžeme začít jinak, než popisem funkce *NIntegrate*. Úplný výklad týkající se dostupných možností funkce *NIntegrate* lze nalézt v nápovědě k programu pod heslem *"Advanced Numerical Integration In Mathematica"*. Stejný text [W08] čítající okolo 150 stran je dostupný ve formátu pdf také na webu. Popíšeme jen nejjednodušší postupy v něm uvedené.

NIntegrate pracuje s metodami, které lze rozdělit na deterministické a Monte Carlo. My se samozřejmě ve výkladu omezíme jen na metody MC. Dále dělíme strategie metod na adaptivní a neadaptivní. Strategie neadaptivní se snaží dosáhnout požadované přesnosti zvyšováním počtu simulací v celém *regionu* (definičním oboru) funkce, naproti tomu adaptivní metody vyhledávají problematické oblasti, region vhodně stratifikují a rozdělí své prostředky nerovnoměrně. Obě strategie mohou k urychlení konvergence využívat symbolický preprocessing, tzn. vlastní aplikaci metody předchází převedení na výhodnější analytické vyjádření.

Použití neadaptivní metody MC, resp. QMC, zajistíme volbou *Method→"MonteCarlo"*, resp. *Method→ "QuasiMonteCarlo"*. Adaptivní metoda MC, resp. QMC, se použije při volbě *Method→ "AdaptiveMonteCarlo"*, resp. *Method→"AdaptiveQuasiMonteCarlo"*. Další speciální volby pro dané metody uvádíme do složených závorek. Podívejme se na příklady.

```
NIntegrate[1/Sqrt[x], {x, 0.01, 1},
 Method → {
   "MonteCarlo",
   "SymbolicProcessing" → 2,
   "RandomSeed" → 1 }]
```

```
1.78276
```

Volba *"SymbolicProcessing"* omezuje zmiňovaný symbolický preprocessing na stanovený počet vteřin a volba *"RandomSeed"* udává použité semínko při generování dat. Dále ukažme, jak provést pevnou stratifikaci v jednotlivých souřadnicích.

```
NIntegrate[x^2 + y^2,
 {x, 0, 1/3, 2/3, 1},
 {y, 0, 1/3, 2/4, 3/4, 1},
 Method → "MonteCarlo"]
```

```
0.672715
```

Pokud nám stačí v každé souřadnici ekvidistantní dělení, zadáme jen počet subintervalů na každé ose volbou *Partitioning*.

```
NIntegrate[x^2 + y^2, {x, 0, 1}, {y, 0, 1},
 Method → {
   "MonteCarlo",
   "Partitioning" → {3, 4}}]
```

```
0.659524
```

Pro srovnávání může být také užitečné stanovit maximální rozsah dat volbou *MaxPoints*.

Závěrem popišme několik obecných voleb pro numerické integrování souvisejících s přesností výpočtu. Pro detailnější popis stačí hledat v nápovědě téma *"Precision & Accuracy Control"*. Volba *WorkingPrecision* změní v daném výpočtu přesnost aritmetiky s čísly, neboli určí maximální počet číslic, které (ve sledu za sebou počínaje nejvyšším řádem) budou reprezentovat číslo v paměti. Pokud *Mathematica* vrátí výsledek s poznámkou, že metoda nekonverguje, pomáhá v některých případech zvýšit tuto hodnotu.

Nastavením *PrecisionGoal*→*p* se *Mathematica* pokusí zajistit, aby relativní chyba výsledku byla menší než 10^{-p}, tedy zhruba řečeno chceme, aby se chyba projevila až v *p*-té číslici v nalezeném výsledku. Nastavením *PrecisionGoal*→∞ nebude brán zřetel na aritmetickou přesnost, tudíž dosažením kterékoliv hranice aritmetické přesnosti se výpočet nezastaví a kritériem ukončujícím výpočet bude volba uvedená níže.

Asi nejstriktnější podmínkou je volba *AccuracyGoal*→*n*, opět lze volit i ∞, rozhodující o hodnotě absolutní chyby výsledku. Výpočet se ukončí v případě, že předpokládaná hodnota chyby výsledku je menší než 10^{-n}, neboli dosáhneme přesnosti na *n* desetinných míst.

Zdůrazněme, že uvedené odhady chyby výsledku jsou pouze orientační, a že přesnost výsledku podmiňuje hlavně úloha. Na následujícím příkladu ukážeme, že k dosažení přesnosti na 3 desetinná místa nestačí u některých metod ani extrémní požadavky na přesnost.

```
NIntegrate[1/Sqrt[x], {x, 0.01, 1},
 Method → "MonteCarlo",
 AccuracyGoal → 3,
 PrecisionGoal → ∞,
 WorkingPrecision → 100,
 MaxPoints → 1 000 000]
```

```
NIntegrate::maxp :
  The integral failed to converge after 1000100 integrand evaluations. NIntegrate obtained
      1.800293661959827113757246872711918352540731787645820378899⋯
    610038996100389961003899610038996100389961`100. and
      0.001148592734427238292092158327638647962871419133455145318⋯
    794456329197206535126040058236555639692189006`100.
      for the integral and error estimates. ≫
```

```
1.800293661959827113757246872711918352540731787645820378899610038996100389961003899610⋯
 038996100389961003899610038996
```

6.2 Pravděpodobnost ztráty a hodnota v riziku

Označme S_t d-rozměrný vektor cen na trhu v čase t. Za čas Δt se ceny změní o $\Delta S_t = S_{t+\Delta t} - S_t$ a určují výnosy $R_{\Delta t}$, tj. relativní změny cen za období Δt, vztahem

$$R_{\Delta t} = \frac{\Delta S_t}{S_t} = \frac{S_{t+\Delta t} - S_t}{S_t}.$$

Investici na trhu v čase t udává d-rozměrný vektor dílčích investic do aktiv s cenami S_t zvaný portfolio. Součet prvků tohoto vektoru, tedy celkovou investici na trhu, označujeme jako hodnotu portfolia V_t. Zisk za období Δt plynoucí z investovaného portfolia v čase t je potom $\Delta V_t = V_{t+\Delta t} - V_t$. Zisku s opačným znaménkem říkáme ztráta, označíme ji $L = -\Delta V_t$. Definujme v *Mathematice* ztrátu L za období Δt jako funkci závislou na portfoliu v čase t a výnosech $R_{\Delta t}$.

```
L[portfolio_, R_] := Total[portfolio - (1 + R) * portfolio]
```

Uvažujme, že ztráta L za období Δt je náhodná veličina charakterizovaná spojitou distribuční funkcí $F_L(x) = \mathrm{P}(L < x)$, $x \in \mathcal{R}$. *Hodnotou v riziku* na úrovni spolehlivosti $1 - \alpha$ potom rozumíme číslo x_α splňující $F_L(x_\alpha) = 1 - \alpha$, ekvivalentně $\mathrm{P}(L > x_\alpha) = \alpha$. Jde tedy o kvantil pravděpodobnostního rozdělení ztráty L.

Předpokládejme, že výnosy ΔR_t za období Δt mají d-rozměrné normální rozdělení $\mathcal{N}(\mu, \Sigma)$, a chceme odhadnout pravděpodobnost, že ztráta L prekročí hodnotu x, pro nějaké dané investované portfolio. V kapitole o generování dat jsme ukázali, jak generovat data z $\mathcal{N}(\mu, \Sigma)$. Aplikací vestavěné funkce *Quantile* na tato data transformovaná výše definovanou funkcí L dostaneme ihned hodnotu v riziku x_α. Funkce *Quantile* aplikovaná na data totiž počítá výběrový kvantil, tzn. kvantil získaný z empirické distribuční funkce L. Pravděpodobnost ztráty odhadneme zřejmě jako

$$\mathrm{P}(L > x) \approx \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{(L_i \leq x)},$$

kde L_i jsou simulované hodnoty ztráty. Jde vlastně o hodnotu empirické distribuční funkce L v bodě x. Podívejme se na konkrétní příklad. Uvažujme daný vektor trendů μ, kovariančí matici mezi výnosy Σ, nějaké rozumné portfolio, úroveň spolehlivosti $1 - \alpha$ a počet simulací n. Spočítáme nejdříve hodnotu v riziku x_α a ověříme, že pravděpodobnost ztráty vyšší než x_α je skutečně α.

```
Remove[x]
```

```
n = 10 000; μ = {0.02, 0.01, 0.0}; Σ = ( 0.03   0.02   0.00
                                         0.02   0.03  -0.01
                                         0.00  -0.01   0.03 );
```

```
portfolio = {1/4, 1/4, 1/2};
α = 0.05;
```

```
R = Map[μ + CholeskyDecomposition[Σ].# & ,
    RandomReal[NormalDistribution[0, 1], {n, Length[Σ]}]];
distL = Map[L[portfolio, #] &, R];
```

```
x = Quantile[distL, 1 - α]
n^-1 Total[BinCounts[distL, {{x, ∞}}]] // N
```

```
0.147911
```

```
0.0501
```

Můžeme si všimnout zajímavosti počítání výběrového kvantilu x, který rozhoduje o tom, jestli se poslední hodnota bude lišit od α o n^{-1}. Toto posunutí není pravidlem, například pro hodnoty $\alpha > 0.7$ bychom v této úloze dostali pravděpodobnost ztráty (větší než x) přesně α.

■ 6.3 Současná hodnota evropské CALL opce

Následující úloha nám poslouží jako vhodná ukázka použití metody Monte Carlo, přestože existuje její explicitní řešení. Půjde nám o odhad očekávané současné hodnoty výplatní funkce evropské CALL opce, a tedy odhad jednorozměrného integrálu, který jsme probrali podrobněji.

Evropská opce CALL (resp. PUT) dává jejímu držiteli právo za smluvenou realizační cenu K koupit (resp. prodat) v čase T dohodnuté aktivum, jeho cenu v čase t označme S_t. Bude-li v čase T cena aktiva S_T menší než K, držitel CALL opce může aktivum za tuto cenu nakoupit na trhu a obratem uplatnit CALL opci, takže jeho zisk v čase T bude vždy $(S_T - K)^+ = max\{0, S_T - K\}$. V modelech se spojitým časem můžeme tuto výplatní funkci (funkci hodnot S_T) diskontovat do současnosti t_0 faktorem $e^{r(t_0-T)}$, kde r je běžná intenzita úrokování konstantní v období (t_0,T).

Nyní uvažujme stochastický model, kde se výnosy z ceny aktiva mění náhodně a spojitě v čase t podle vztahu

$$\frac{dS_t}{S_t} = \mu\, dt + \sigma\, dW(t),$$

kde W je normovaný Wienerův proces. Parametr μ zastupuje střední míru návratnosti (mean rate of return) a σ volatilitu daného aktiva, tyto veličiny v praxi máme kótované. Volbou $\mu = r$, tj. nahrazením μ běžnou intenzitou úrokování, dostaneme tzv. *rizikově neutrální model* ceny aktiva. Řešením uvedené diferenciální rovnice je

$$S_T = S_{t_0}\, e^{\left(r-\frac{1}{2}\sigma^2\right)(T-t_0)+\sigma \mathrm{W}(T-t_0)}.$$

Přírůstek Wienerova procesu z času t_0 do T má normální rozdělení $\mathcal{N}(0,T-t_0)$, vztah proto můžeme přepsat jako

$$S_T = S_{t_0}\, e^{\left(r-\frac{1}{2}\sigma^2\right)(T-t_0)+\sigma\sqrt{T-t_0}\,Z},$$

kde náhodná veličina Z vniklá normováním W má normované normální rozdělení $\mathcal{N}(0,1)$. Současnou hodnotu $S(t_0)$ považujeme za známou a logaritmováním předchozího vztahu dostaneme veličinu s normálním rozdělením, neboli náhodná veličina $S(T)$ má v uvedeném modelu logaritmicko-normální rozdělení. Snadno se nahlédnou parametry polohy i měřítka tohoto rozdělení

$$S_T \sim \mathcal{LN}\left(\log S_{t_0} + \left(r - \frac{1}{2}\sigma^2\right)(T-t_0),\ \sigma^2(T-t_0)\right).$$

Naše očekávání vyjadřujeme střední hodnotou, proto očekávaná současná hodnota výplatní funkce je $\mathrm{E}\left(e^{r(t_0-T)}(S_T-K)^+\right)$, tj. integrál vzhledem k logaritmicko-normální hustotě rozdělení S_T. Dále budeme bez újmy na obecnosti uvažovat, že $t_0 = 0$, protože v použitém modelu lze vyjadřovat přírůstky jako funkce rozdílu argumentů $T - t_0$, takže T bude zároveň doba od současnosti do času možné realizace opce.

Užitím Blackovy-Scholesovy formule

$$BS(S, \sigma, T, r, K) = S\,\Phi\left(\frac{\log\frac{S}{K} + \left(r + \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}}\right) - e^{-rT}\,K\,\Phi\left(\frac{\log\frac{S}{K} + \left(r - \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}}\right),$$

kde Φ je distribuční funkce rozdělení $\mathcal{N}(0,1)$, lze uvedený integrál vyčíslit přesně a pro obecné t_0 tedy platí

$$\mathrm{E}\left(e^{r(t_0-T)}(S(T)-K)^+\right) = BS(S_{t_0}, \sigma, T - t_0, r, K).$$

Pro pozdější použití zapišme v *Mathematice* funkci *BS* a definujme distribuci S_T.

```
Remove[Φ, BS]

Φ[x_] := CDF[NormalDistribution[0, 1], x]

BS[S_, σ_, T_, r_, K_] :=
 S Φ[(Log[S/K] + (r + σ^2/2) T)/(σ Sqrt[T])] - e^(-r T) K Φ[(Log[S/K] + (r - σ^2/2) T)/(σ Sqrt[T])]

distST[S_, σ_, T_, r_] :=
 LogNormalDistribution[Log[S] + (r - σ^2/2) T, σ Sqrt[T]]
```

Nyní použijme k odhadu integrálu metodu Monte Carlo. Protože funkce bude obsahovat několik voleb, začněme krátkou specifikací. Funkce *MCMBS* bude obsahovat stejné parametry jako funkce *BS* a navíc buď parametr *n* rozsahu dat anebo rovnou seznam generovaných dat. Dále můžeme funkci na vstupu změnit několik voleb. Volbou *"TypeData"→"Uniform"* dáme najevo, že seznam generovaných vstupních dat je realizací náhodného výběru z rovnoměrného rozdělení $\mathcal{U}([0,1])$. Funkci ale můžeme dodat data z normovaného normálního rozdělení $\mathcal{N}(0,1)$, potom nastavujeme *"TypeData"→"Normal"*, nebo dodáme rovnou realizovaný náhodný výběr z rozdělení $S(T)$, v takovém případě nastavíme *"TypeData"→"LogNormal",* případně neuvádíme volbu vůbec, protože bude takto definována implicitně (defaultně). Volbou *"CI"→True* zajistíme, že funkce vypíše i interval spolehlivosti pro výsledný odhad, přičemž můžeme měnit hladinu spolehlivosti nastavením hodnoty *ConfidenceLevel* a také rozdělení, na kterém je intervalový odhad založen. Můžeme vybrat buď intervalový odhad z normální distribuce volbou *"TypeCI"→"Normal"*, nebo ze Studentovy t-distribuce volbou *"TypeCI"→"StudentT"*.

```
Remove[MCMBS]
Needs["HypothesisTesting`"]

Options[MCMBS] = {"TypeData" → "LogNormal", "CI" → False,
    "TypeCI" → "StudentT", ConfidenceLevel → 95 / 100};

MCMBS[S_, σ_, T_, r_, K_, Data_List, o___?OptionQ] := Module[
  {n = Length[Data], distEstim, distS, distNorm, M, s, interval,
   ci, typeCI, typeData, α},
  {ci, typeCI, typeData, α} =
   {"CI", "TypeCI", "TypeData", ConfidenceLevel} /.
    Flatten[{o, Options[MCMBS]}];
  Which[
   typeData == "LogNormal",
   distEstim = Map[e^(-r T) Max[0, # - K] &, Data],
   typeData == "Normal",
   distS = Map[S Exp[(r - 1/2 σ^2) T + σ Sqrt[T] #] &, Data];
   distEstim = Map[e^(-r T) Max[0, # - K] &, distS],
   typeData == "Uniform",
   distNorm = Map[Sqrt[2] InverseErf[-1 + 2 #] &, Data];
   distS = Map[S Exp[(r - 1/2 σ^2) T + σ Sqrt[T] #] &, distNorm];
   distEstim = Map[e^(-r T) Max[0, # - K] &, distS]];
 If[ci,
   M = Mean[distEstim];
   s = Sqrt[Variance[distEstim]];
   Which[
    typeCI == "StudentT",
    interval = StudentTCI[M, s/Sqrt[n], n - 1, ConfidenceLevel → α],
    typeCI == "Normal",
    interval = NormalCI[M, s/Sqrt[n], ConfidenceLevel → α]];
   TableForm[{{M, ToString[interval]}},
    TableHeadings → {None, {"Estimation",
       ToString[100 α] <> "% Confidence Interval"}}],
   Mean[C]]]
```

```
MCMBS[S_, σ_, T_, r_, K_, n_Integer, o___?OptionQ] :=
 MCMBS[S, σ, T, r, K, RandomReal[
   LogNormalDistribution[Log[S] + (r - 1/2 σ^2) T, σ Sqrt[T]], n], o]
```

Následuje vlastní použití funkce. Na výsledných výstupech si povšimněme především časových náročností. Funkce je koncipována pro lepší přehlednost tak, že jednotlivé transformace distribucí se provádí postupně (což zvyšuje jak paměťovou, tak i časovou náročnost). Můžeme si proto všimnout, že například transformace dat s rozsahem $n = 10\,000$ z rovnoměrného rozdělení na normální zabírá bezmála 2 vteřiny. Kdybychom chtěli snížit časové nároky, ve funkci bychom raději aplikovali přímo složení jednotlivých transformací.

```
S0 = 1; σ = 0.5; T = 2; r = 0.07; K = 1;

Timing[MCMBS[S0, σ, T, r, K, 10 000, "CI" → True]]

Data = RandomReal[NormalDistribution[0, 1], 10 000];
Timing[MCMBS[S0, σ, T, r, K, Data,
  "CI" → True, "TypeData" → "Normal"]]

Data = RandomReal[{0, 1}, 10 000];
Timing[MCMBS[S0, σ, T, r, K, Data,
  "CI" → True, "TypeData" → "Uniform"]]

BS[S0, σ, T, r, K] // N
```

```
{0.062, Estimation  95% Confidence Interval
        0.323594    {0.310196, 0.336992}}

{0.078, Estimation  95% Confidence Interval
        0.33679     {0.322578, 0.351001}}

{2.516, Estimation  95% Confidence Interval
        0.324231    {0.311111, 0.33735}}

0.328427
```

■ 6.4 Oceňování amerických opcí

Americká opce se od evropské liší v možnosti jejího uplatnění kdykoliv před časem vypršení T. Opět bez újmy na obecnosti vztahujme naše úvahy k současnosti $t_0 = 0$. Hodnota americké opce tak závisí na průběhu ceny aktiva S_t v celém časovém intervalu $[0,T]$.

Existují různé modely vývoje ceny a řada postupů, jak hodnotu opce odhadnout. Často se uvažují Markovovy procesy, ve kterých nadcházející vývoj ceny (pravděpodobnostní rozdělení ceny v čase $t + \Delta t$) závisí pouze na ceně v aktuálním okamžiku t (nezávisí na předcházejícím vývoji ceny). Například Blackův-Scholesův stochastický model z předchozí podkapitoly splňuje markovskou vlastnost.

Pro účely numerického počítání je nutné model pracující se spojitým časem diskretizovat. Proto se omezujeme pouze na opce s možností uplatnění v některém z okamžiků $t_1, t_2, \ldots, t_m$, kde $0 = t_0 < t_1 < \ldots < t_m = T$. V mnoha případech je takové omezení přímo součástí opční smlouvy, potom opci označujeme jako *bermudskou*, protože leží někde mezi opcí americkou a evropskou. Každého asi napadne, že zvyšováním počtu m těchto okamžiků lze stejné postupy použít i pro odhady cen amerických opcí. Složitost některých postupů však s rostoucím m roste exponenciálně, což se týká také metody binomických stromů, kterou dále popíšeme.

Budeme předpokládat, že na intervalu $[0,T]$ rozděleným ekvidistantním dělením na subintervaly $[t_{i-1}, t_i]$, $i = 1, \ldots, m$, se cena aktiva řídí dynamikou popsanou u evropské opce, takže pro popis pohybu ceny použijem znovu parametry r, σ a změnu ceny za období Δt při počáteční ceně S_{i-1} určuje logaritmicko-normální rozdělení

$$S_{t_i} \sim \mathcal{LN}\left(\log S_{t_{i-1}} + \left(r - \tfrac{1}{2}\sigma^2\right)\Delta t,\ \sigma^2 \Delta t\right),\ i = 1, \ldots, m\ ,$$

a tedy logaritmus této náhodné veličiny má normální rozdělení se stejnými parametry

$$ln\, S_{t_i} \sim \mathcal{N}\left(\log S_{t_{i-1}} + \left(r - \tfrac{1}{2}\sigma^2\right)\Delta t,\ \sigma^2 \Delta t\right),\ i = 1, \ldots, m\ .$$

Protože v kapitole o generování dat jsme si vytvořili funkci *GenerateNormalTree*, která generuje stromy využitím normálního rozdělení, budeme simulovat logaritmy cen. Mějme proto napaměti, že za počáteční hodnotu volíme $ln\, S_{t_0}$ a parametr driftu a časový posun jako

$$\mu = \left(r - \tfrac{1}{2}\sigma^2\right)\Delta t\ ,\ \ \Delta t = \tfrac{T}{m}.$$

Funkci ještě předáváme parametr směrodatné odchylky rozdělení $\sigma\sqrt{\Delta t}$, a zvolíme libovolně parametr větvení stromu b. K získání stromu skutečných cen pocházejících z logaritmicko-normálního rozdělení stačí mapovat exponenciálu na všechny prvky stromu s cenami logaritmickými.

Podobně jako jsme deklarovali odhadující funkce dříve, i nyní u funkce *PriceOfAmericanOption* ponecháme možnost výběru mezi předáním standartních parametrů (S_{t_0}, σ, T, r, K, b, m), které se použijí pro tvorbu dat, anebo předáme funkci data již vytvořená, potom stačí předat parametry (Δt, r, K, *Data*). Parametr K vyjadřuje stejně jako v předchozí podkapitole realizační cenu. Funkci *PriceOfAmericanOption* bude možné použít k oceňování PUT i CALL opcí, protože výplatní funkci *Payoff* zavedeme odděleně. Protože rekurzivní prostup pro samotné hodnocení opce není úplně triviální, popíšeme byť stručný zápis o něco podrobněji.

Nejprve uvažme, jakou hodnotu by pro nás měla opce k datu expirace při ceně aktiva S_T a realizační ceně K. Jistě by se jednalo o přímou aplikaci výplatní funkce na cenu S_T, kterou nemusíme diskontovat. Dále se zamysleme, jak bychom hodnotili opci v předposledním možném okamžiku uplatnění t_{m-1}. Známe konkrétní hodnotu aktiva $S_{t_{m-1}}$ a předpokládáme, že se cena za dobu Δt posune s ohledem k modelovému rozdělení. Cenu S_{t_m} proto považujeme za náhodnou veličinu. Nyní se rozhodujeme, jestli uplatníme opci okamžitě, anebo za dobu Δt. Z okamžitého uplatnění pramení zisk spočitatelný opět přímou aplikací výplatní funkce na $S_{t_{m-1}}$. Předpokládaný zisk z uplatnění opce v čase t_m spočítáme analogicky jako u evropské opce. Nejprve na všechny možné hodnoty aplikujeme výplatní funkci, poté střední hodnotu (vyjádření očekávání) a nakonec částku diskontujeme přes období Δt pro srovnání v současné hodnotě peněz. Možnými hodnotami máme v modelu na mysli

celý nosič rozdělení S_{t_m}, při počítání metodami MC potom generované hodnoty z tohoto rozdělení a zde ještě konkrétněji následovníky $S_{t_{m-1}}$ v generovaném stromu. Střední hodnotu samozřejmě nahrazujeme v metodách MC výběrovým průměrem a diskontování provádíme s ohledem ke spojitému modelu s konstantní intenzitou r, i když v limitě pro $m \to \infty$ by bylo ekvivalentní použití i složeného diskontování. Popsaný způsob nám nabízí možnost srovnání zisku plynoucího z okamžitého a odloženého uplatnění opce v hodnotě peněz v okamžiku t_{m-1}. Při hodnocení ceny opce samozřejmě vybíráme vždy ten vyšší zisk, protože se nám naskýtají obě možnosti. Podstata algoritmu spočívá ve zpětné indukci, kdy postupně od listí stromu v čase T takto ohodnocujeme opci v časech o Δt nižších pro všechny generované alternativy až ke kořeni stromu v čase $t_0 = 0$. K procházení stromu (resp. tvorbě nového) použijeme opakované použití jednoduchých pravidel podobně jako při jeho generování. Podotkněme nakonec, že paměťové nároky následujícího algoritmu nejsou zdaleka optimální. Pro ekvivalentní postup není vůbec nutné uchovávat v paměti celý strom.

```
Remove[PriceOfAmericanOption, Payoff]

Payoff[S_, K_] := Max[0, S - K] (* CALL option *)

PriceOfAmericanOption[S_, σ_, T_, r_, K_, b_Integer, m_Integer] :=
 PriceOfAmericanOption[T/m, r, K, Map[Exp,
   GenerateNormalTree[m, b, (r - σ^2/2) T/m, Sqrt[T/m] σ, Log[S]], {-1}]]

PriceOfAmericanOption[Δt_, r_,
  K_, h_Real, v_?(VectorQ[#, NumericQ] &)] :=
 K + Max[Payoff[h, K], ℯ^(-r Δt) Mean[Map[Payoff[#, K] &, v]]]

PriceOfAmericanOption[Δt_, r_, K_, Data_List] := Module[{h, v},
  (Data //. {h_Real, v_?(VectorQ[#, NumericQ] &)} →
      PriceOfAmericanOption[Δt, r, K, h, v]) - K]
```

Při aplikaci metody nejprve zvolíme pro kontrolu stejné parametry jako u evropské CALL opce z předchozí podkapitoly a také nepoužijeme žádné dělení intervalu $[0,T]$ volbou $m = 1$. Měli bychom tedy v průměru obdržet stejné výsledky jako dříve. Potom použijeme dělení na $m = 4$ subintervaly a parametr větvení $b = 20$, takže generovaný strom bude obsahovat $20^4 = 160\,000$ listů (možných cen S_T). Mělo by se (aspoň v průměru) projevit, že cena americké opce je samozřejmě vždy vyšší než cena evropské se stejnými parametry.

```
S0 = 1; σ = 0.5; T = 2; r = 0.07; K = 1; b = 10 000; m = 1;

PriceOfAmericanOption[S0, σ, T, r, K, b, m]
0.323775
```

```
b = 20; m = 4;

PriceOfAmericanOption[S0, σ, T, r, K, b, m]
0.370424
```

V popsané metodě lze prakticky využít jen malé hodnoty *m*. Nízká hodnota parametru *b* také způsobuje značnou vychýlenost odhadu. Teoretické vlastnosti postupu sice zaručují, že s rostoucím *m* a *b* konverguje střední hodnota odhadu ke skutečné (modelové) hodnotě opce, nicméně rychlost této kovergence není valná. Prakticky lze metodu použít až při kombinaci s méně vychýlenou analogickou metodou. Protože by toto téma vydalo nejméně za další kapitolu, odkážeme čtenáře na [G04], kde může najít nejen tuto, ale mnoho dalších metod pro oceňování amerických opcí.

Seznam použité literatury

[D05] V. Dupač, M. Hušková: *Pravděpodobnost a matematická statistika*. Karolinum, 2005.

[G04] P. Glasserman: *Monte Carlo Methods in Financial Engineering*. Springer, 2004.

[H82] J. Hurt: *Simulační metody*. SPN, 1982.

[P07] Z. Prášková, P. Lachout: *Základy náhodných procesů I & II*. Karolinum, 2007.

[S02] R. Seydel: *Tools for Computional Finance*. Springer, 2002.

[U03] G. Uhlmann: *Inside Out*. Cambridge University Press, 2003.

[W08A] Wolfram Research: *Advanced Numerical Integration In Mathematica*. 2008.

[W08R] Wolfram Research: *Random Number Generation*. 2008.

Průvodce k programu *Mathematica* lze nalézt také na webových stránkách
http://www.wolfram.com/learningcenter/tutorialcollection/