POČÍTAČOVÁ FYZIKA I

Rudolf Hrach

PF UJEP Ústí nad Labem

2003

Obsah

Kapitola 1

ÚVOD

Počítačová fyzika patří mezi nové a prudce se rozvíjející směry vědeckého bádání. Název, jenž vznikl spojením dvou pojmů – fyzika a počítač – je výstižným popisem jejího zaměření. Na rozdíl od jiných směrů vědecké práce, kde charakterizujícím faktorem je *objekt studia* (jako např. fyzika pevných látek, fyzika polovodičů, fyzika plazmatu, apod.), v případě počítačové fyziky je rozhodující *metodika řešení* fyzikálního problému. Tím dochází k omezování již dlouho probíhajícího a ne vždy vítaného trendu stále hlubší specializace (např. pevná látka → polovodič → amorfní polovodič → ...).

Z hlediska metodického se vědecká práce ve fyzice (a nejen tam) již dlouho dělí na dva základní směry – na ***experimentální*** a ***teoretický***. Experimentální fyzika produkuje data, dává podněty pro vytváření teorií a umožňuje tyto teorie testovat. Teoretická fyzika interpretuje a zobecňuje experimentální poznatky a navrhuje další experimenty sloužící jednak k prohloubení znalostí ve studovaných oblastech a případně i dává podněty pro studium oblastí nových.

Tato symbioza obou přístupů velmi úspěšně probíhá po mnoho staletí, resp. tisíciletí. V posledních několika desetiletích však začíná být narušována, lépe řečeno doplňována, právě ***počítačovou fyzikou***. Ta se snaží popsanou dvoučlennou spolupráci experimentální a teoretické fyziky změnit na rovnocennou spolupráci tří partnerů: experimentální, teoretické a počítačové fyziky. Vedle již popsaných vazeb historických směrů tak přibývají další vazby. Experimentální fyzika generuje data využívaná jak teoretickou tak i počítačovou fyzikou, počítačová fyzika naopak pro experimentální fyziku provádí analýzu těchto dat, zabezpečuje řízení experimentálních zařízení, vytváří modely reálných procesů a na jejich základě analyzuje experimentální data a navrhuje nové experimenty. Podobná vazba se vytváří i mezi počítačovou a teoretickou fyzikou: teoretická fyzika poskytuje rovnice pro řešení metodami počítačové fyziky a naopak interpretuje výsledky počítačovými metodami zís-

kané, počítačová fyzika pak poskytuje teoretické fyzice pomoc při provádění rozsáhlých výpočtů a podobně jako experimentální fyzika dává prostřednictvím tzv. počítačových experimentů podněty pro vytváření nových teorií a pomáhá při jejich testování. Takto nastíněná spolupráce tří rovnocenných metodik řešení fyzikálních problémů je v současné době pro počítačovou fyziku zatím ještě poněkud ambiciozním plánem, s rozvojem hardwarových a softwarových prostředků se však stává stále reálnější.

Co to vlastně je počítačová fyzika? Přesná definice tohoto vědního oboru je velmi obtížná, ne-li nemožná. Stručně řečeno, jsou to takové postupy při řešení fyzikálních problémů, při kterých počítač hraje podstatnou roli. Důležité je slovo „podstatnou“, protože v současné době najdeme počítač v každé laboratoři, kde se užívá při drobnějších výpočtech, psaní textů (publikací, disertačních prací), sběru dat, atd. Určitě sem patří např. počítačové simulace nebo symbolické manipulace, částečně i automatizace experimentu, apod. S rozvojem hardware se vytvářejí i nové oblasti počítačové fyziky jako např. práce v počítačových sítích nebo studium paralelních procesů. Počítačová fyzika jednak poskytuje výstup výsledků do jednotlivých fyzikálních disciplin (a zde často tyto postupy bývají považovány za součást příslušných oborů) a jednak vytváří nové metody práce, čímž dále počítačovou fyziku jako disciplínu rozvíjí. Hranice oboru jsou z tohoto důvodu mlhavé a neustále se mění.

Pro potřeby tohoto studijního textu byla počítačová fyzika rozdělena do následujících základních směrů, kterým se budeme postupně věnovat:

- Počítačové modelování
- Počítačová grafika a vizualizace
- Zpracování obrazu
- Integrální transformace

Z těchto oblastí má v počítačové fyzice výsadní postavení *počítačové modelování*, což je technika používaná fyziky různých specializací nejčastěji, neboť jim poskytuje často neocenitelnou pomoc při experimentálním i teoretickém studiu nejrůznějších fyzikálních jevů a procesů. Je také nejvíce rozpracována a dělí se na řadu dílčích metodik, které jsou často známé i mimo rámec počítačové fyziky - např. metoda Monte Carlo. Proto se i my budeme této oblasti počítačové fyziky věnovat podrobněji a bude tvořit těžiště prvního dílu tohoto studijního textu. Ostatní tři vyjmenované směry počítačové fyziky budou tvořit náplň druhého dílu studijního textu. Jelikož vyjmenovanými základními směry počítačová fyzika zdaleka nekončí, v závěru druhého dílu

se stručně zmíníme i o dalších metodikách počítačové fyziky, které buď patří do počítačové fyziky jen okrajově (např. Řízení experimentu) a nebo jsou tak rozsáhlé a obtížné, že překračují rámec tohoto studijního textu (Symbolické manipulace, Teorie waveletových funkcí, atd.).

I když počítačová fyzika patří mezi nejmladší vědecké disciplíny, již se rozvinula do takové šíře a dosáhla tolika výsledků, že se začíná vnitřně dělit. Stále častěji se hovoří o

- klasické počítačové fyzice
- moderní počítačové fyzice.

Pod pojem *moderní počítačová fyzika* se zahrnují nové progresívní směry, které se objevily v posledních letech buď přímo v rámci počítačové fyziky a nebo byly jako aplikace převzaty z jiných vědeckých oblastí (nejčastěji z matematiky a informatiky). Do moderní počítačové fyziky v současné době patří neuronové sítě (resp. jejich aplikace ve fyzice), fuzzy logika a evoluční programování (genetické algoritmy), přičemž tento seznam je nadále otevřený. Z tohoto hlediska předložený studijní text se věnuje klasické počítačové fyzice, pouze v závěru druhého dílu budou velmi stručně nastíněny problémy řešené postupy moderní počítačové fyziky.

> Závěrem ještě jednu poznámku ke členění textu. Základní informace v textu obsažené jsou určené pro bakalářskou úroveň studia fyziky. Pokud to ale bude vhodné, bude výklad doplněn tak, aby se zájemce (nebo případně posluchač magisterského studia) dozvěděl o příslušné problematice více. Rozšiřující informace v tomto studijním textu budou od základního výkladu odděleny graficky – budou psány drobnějším písmem a odsazeny od okrajů stránky.

K účelu doplnění a rozšíření znalostí získaných při přednáškách slouží též seznam literatury uvedený na konci obou dílů studijního textu. Naneštěstí neexistuje dostatečné množství literatury k jednotlivým partiím počítačové fyziky v češtině, proto je většina uváděných knih v angličtině a v knihovnách jsou jen v omezeném počtu. Pro bakalářské studium však čtení rozšiřující literatury není nezbytné (i když samozřejmě je užitečné).

Každá další kapitola obou dílů studijního textu bude zakončena sekcí *Shrnutí*, v níž budou uvedeny nejdůležitější poznatky v příslušné kapitole popsané. Tyto poznatky budou též vyžadovány u zkoušek – dílčích i státních. Dále budou v některých kapitolách uvedeny příklady určené k procvičování látky v kapitole popsané. Některé z těchto příkladů budou již vyřešeny, většina však bude určena k samostudiu, případně k procvičení v rámci organizované výuky.

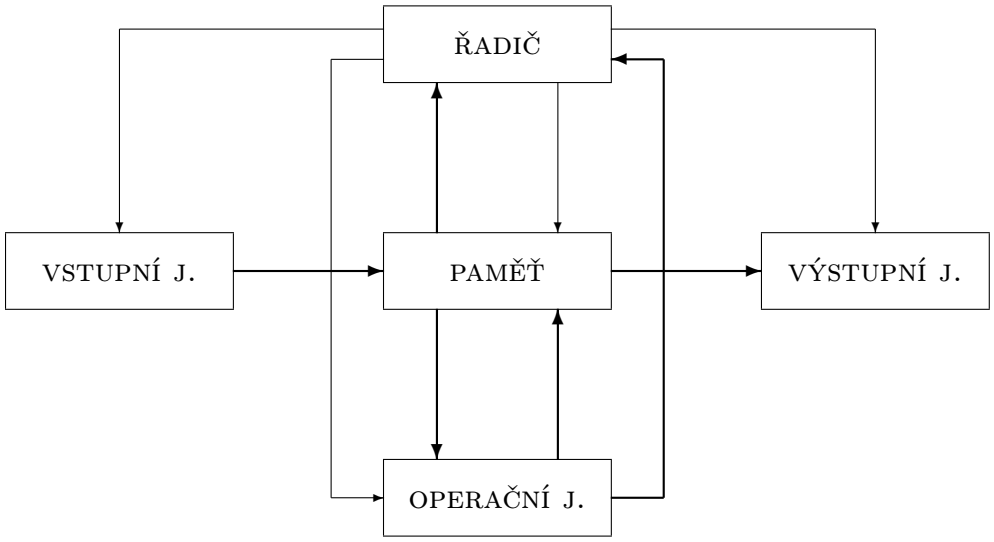
Kapitola 2

HARDWAROVÉ A SOFTWAROVÉ ZÁKLADY POČÍTAČOVÉ FYZIKY

Pro použití počítačové fyziky k řešení fyzikálních problémů musíme mít odpovídající výbavu – jak výpočetní techniku (hardware), tak i prostředky softwarové (vhodný programovací jazyk, počítač vybavený příslušným operačním systémem, ...). Především však musíme mít patřičné znalosti, jak tyto prostředky použít k vyřešení našeho problému. Poslední část výbavy, znalosti, by měly být obsaženy v tomto studijním textu, hardwarové a softwarové prostředky musí poskytnout příslušné pracoviště. A co bychom na něm měli požadovat?

2.1 Hardwarové prostředky

Nebudeme se zde detailně zabývat *prehistorií* výpočetní techniky, i když je velmi zajímavá. Již v raném středověku v Evropě (a v Číně ještě podstatně dříve) byly činěny více či méně úspěšné pokusy o konstrukci umělých bytostí – homunkulů, mechanických písařů, šachistů, atd. – dobové zápisy nám však nedovolují jednoznačně určit, co bylo reálně vytvořeno a co byl již tehdy podvod, případně co je pouhá fáma. Není proto možno rozhodnout, kdy první současné výpočetní technice podobné zařízení vzniklo. Určitě však se nejedná o století dvacáté, neboť již počátkem 19. století se objevila dvě jména, která do historie výpočetní techniky zcela určitě patří – konstruktér v moderním slova smyslu, profesor matematiky Charles Babbage a první skutečná programátorka světa, dcera lorda Byrona, Augusta Ada Lovelace (po níž byl pojmenován ve své době populární programovací jazyk ADA).



Obrázek 2.1: Von-Neumannovská koncepce počítače. Tok dat je znázorněn silnými čarami, řídící příkazy čarami slabými.

Skutečný rozvoj výpočetní techniky však začal až v první polovině století dvacátého. Tehdy se začaly formovat pojmy, které jsou dnes již tak známé, že je zde není nutno vysvětlovat – bit, byte, výpočetní soustava (dvojková, desítková, osmičková, ...), logické operátory, atd. (pojem algoritmus však zavedla již Ada Lovelace) – především však vznikla tzv. *Von-Neumannova koncepce* počítače, podle níž byly počítače (a mikropočítače) konstruovány od samých počátků až do nedávné doby. Tato koncepce je znázorněna na obr. 2.1. Podle ní se počítač má skládat z několika základních bloků (paměť, řadič, operační jednotka, vstupní a výstupní jednotka), mezi kterými procházejí data i řídící příkazy podle toho jak určuje program.

> Základní postavení má paměť, do které se ukládají jak vstupní data, mezivýsledky a konečné výsledky, tak i program tvořený posloupností jednotlivých příkazů (tím se počítač lišil od tzv. programovatelných kalkulátorů, kde paměti pro data a programy byly zcela odděleny).
>
> Před počátkem výpočtu je program, a případně i vstupní data, načten do paměti. Jednotlivé příkazy programu jsou postupně posílány do řadiče, který je dešifruje a podle potřeby přikazuje ostatním blokům počítače, aby provedly příslušnou činnost – načetly další vstupní data (Vstupní jednotka → Paměť), zajistily výstup výsledků z počítače (Paměť → Výstupní jednotka) a především prováděly aritmetické a logické operace (Paměť, Operační jednotka, Řadič). Konkrétně, paměť pošle operandy do operační jednotky, ta provede příslušnou operaci

a výsledek vrátí jednak do paměti a též do řadiče. Tím může řadič modifikovat další běh programu podle výsledku předchozí operace – podmíněné skoky, atd.

Tato architektura počítače byla základem konstrukce výpočetní techniky po více než padesát let jak u velkých (sálových) počítačů tak později i u mikropočítačů. Během doby samozřejmě doznala určitých změn – nejprve díky hardwarovým omezením u počítačů nulté a první generace (viz dále) a později naopak díky rozvoji mikroprocesorové techniky. Nejvýznamnější jsou asi nahrazení pojmů vstupní a výstupní jednotka pojmem *kanál*, na nějž jsou teprve periférie připojovány (což umožnilo současnou práci více vstupních a výstupních zařízení) a specializace jednotlivých částí paměti (*zásobníková paměť*, *registry*, ...). K tomu se podrobněji vrátíme v druhé části studijního textu v souvislosti s řízením experimentu.

Teprve v posledních letech se začaly konstruovat počítače zcela se vymykající von Neumannově koncepci – paralelní a vektorové systémy, dovolující ve stejném okamžiku zpracovat více informací na rozdíl od sériové činnosti starších systémů. I k této otázce se vrátíme ve druhém dílu studijního textu. (Mimo jiné, koncepci těchto paralelních systémů, které se někdy symbolicky označují jako „Non-Von“, navrhl již v polovině 20. století též John von Neumann).

V následující tabulce si stručně shrneme vývoj výpočetní techniky ve dvacátém století – do tohoto přehledu není započítána mikroprocesorová technika, kterou si podrobněji probereme později, stejně jako současné počítače. Jen pro srovnání – vývoj pokračuje i nadále na bázi integrovaných obvodů a počet prvků v jednom obvodu od počátečních několika desítek prvků se v současnosti blíží ke sto miliónům. Moderní paralelní systémy obsahují takových integrovaných obvodů (mikroprocesorů) až několik set či tisíc, čemuž odpovídá i výpočetní výkon. Podrobněji viz druhý díl studijního textu.

generace	**rok**	**konstr. prvky**	**výkon [op/s]**	**RAM [b]**	**SW**
0	1941	relé	10^1	10^3	stroj. kód
1	1946	elektronky	10^2	10^4	assembler
2	1960	polovodiče	10^3	10^5	FORTRAN
2,5	1964	mikromoduly	10^4	10^6	op. systémy
3	1966	int. obvody	10^5	10^7	sdílení času
3,5	1971	int. obvody	10^6	10^8	virt. paměti
...					

Tabulka 2.1: Hlavní charakteristiky výpočetní techniky.

Časové údaje v tabulce 2.1 uvedené odpovídají době, kdy se příslušná technika začala masově používat, vlastní začátek vždy poněkud předbíhal (např. první počítač osazený tranzistory byl zkonstruován již v roce 1955).

S růstem složitosti výpočetních systémů roste i výpočetní výkon, jehož hodnota se zvyšuje podle empirického Moorova zákona (formulovaného v roce 1964): zdvojnásobuje se každých dvanáct až osmnáct měsíců. Proto od počátku 40. let výkon počítaný v desítkách operací za sekundu vzrostl až na současných téměř neuvěřitelných $35,61 \cdot 10^{12}$ operací za sekundu (a to v pohyblivé řádové čárce, tzv. teraflops). Jedná se o superpočítač Earth Simulator firmy NEC uvedený do provozu v roce 2002 a obsahující 5 134 vektorových procesorů.

2.2 Softwarové prostředky

Při formulování zadání fyzikálního problému na počítači a jeho vyřešení postupy počítačové fyziky musíme mít odpovídající softwarové prostředky, především vhodný *programovací jazyk* a dále prostředky na ovládání procesu výpočtu, tj. *operační systém*.

Vlastním „jazykem“ počítače je tzv. *strojový kód*. Program ve strojovém kódu je tvořen posloupností příkazů, které oznamují řadiči počítače, jakou operaci má provést, s jakými argumenty, co má udělat s výsledkem operace a jak má program pokračovat. Všechny tyto informace jsou uvedeny ve dvojkové soustavě a jejich zápis se liší podle jednotlivých typů počítačů (nebo mikroprocesorů). Příklady zápisu programu ve strojovém kódu mikroprocesorů Intel budou uvedeny ve druhém díle studijního textu.

Jelikož vytváření programu ve strojovém kódu je mimořádně náročné, od počátku rozvoje výpočetní techniky byla snaha tuto činnost programátorům ulehčit – uživatelům ponechat pouze tvůrčí práci a mechanickou činnost přenechat počítači. To vedlo k vytvoření ***programovacích jazyků***.

Mezistavem bylo zavedení tzv. *assemblerů*, tj. jazyků symbolických adres či jazyků symbolických instrukcí. Nejedná se ještě o programovací jazyky v pravém slova smyslu ale jen o zrychlený zápis programu ve strojovém kódu se zachováním jeho základní výhody – maximální rychlosti výpočtu – spolu se snížením námahy na vytváření programů. Proti programování v pravých programovacích jazycích je ale práce v assembleru výrazně náročnější, proto se v současné době používá jen tam, kde je rychlost výpočtu a plná kontrola nad činností počítače nezbytná, tj. především při řízení experimentu (a to ještě zdaleka ne vždy). Ve všech ostatních případech se dává přednost práci ve vyšších programovacích jazycích.

O programovací jazyk se jedná tehdy, když jedné instrukci programu

obecně odpovídá více instrukcí ve strojovém kódu. To má za následek na jedné straně zefektivnění programátorské práce, na druhé straně však to přináší menší efektivitu výsledného kódu. Jelikož prioritou však je snížená námaha programátora, v počítačové fyzice se prakticky výhradně pracuje ve vyšších programovacích jazycích. K tomu, abychom zvolili vhodný jazyk, je dobré znát něco o jejich historii.

Překladač programového jazyka je speciální program, který na vstupu má zápis naší úlohy v symbolice daného jazyka a na výstupu je program ve strojovém kódu příslušného procesoru. Vznik prvních programovacích jazyků je proto spojen s dostatečným rozvojem výpočetní techniky – kapacity paměti, rychlosti procesoru, atd. – aby bylo možno spolu s vlastním programem uživatele zpracovávat i překladač příslušného jazyka. Podle tabulky 2.1 se jedná o přechod mezi první a druhou generací počítačů, někdy kolem roku 1957. Tehdy ve velmi krátkém rozmezí vznikly první tři programovací jazyky, které tvoří základ velkého množství současných jazyků. Jednalo se o jazyky FORTRAN – vhodný především pro vědecko-technické výpočty, ALGOL – pro matematický zápis algoritmů a COBOL – pro ekonomické výpočty. Pak následoval velmi prudký rozvoj, který do naší doby ještě neskončil. Podle posledního seriózně provedeného sčítání, které (podle naší informace) proběhlo v roce 1980, tehdy existovalo již více než 2 000 programovacích jazyků. Jejich současný počet si nedovolujeme ani přibližně odhadnout. Většina z nich je však jen úzce specializovaná, takže většího rozšíření v naší oblasti vědy dosáhlo jen několik z nich.

Programovací jazyky se dělí na různé skupiny podle řady klíčů:

- jazyky strojově orientované a problémově orientované
- jazyky kompilační a interpretační
- jazyky vhodné pro vědecko-technické výpočty, pro ekonomii, ...
- atd.

Pokud se budeme dále věnovat jen programovacím jazykům vhodným z hlediska počítačové fyziky, nebo obecněji jazykům používaným v matematice a fyzice, je třeba uvést tyto jazyky:

Skupina jazyků FORTRAN: Původní jazyk FORTRAN z roku 1957 se přes několik mezistupňů vyvinul v jazyk FORTRAN-66 (dnes již nepoužívaný), v jazyk FORTRAN-77 (již zastaralý, ale na řadě pracovišť dosud používaný, proto je vhodné znát aspoň jeho základy), v jazyk FORTRAN-90 a v zatím poslední FORTRAN-95.

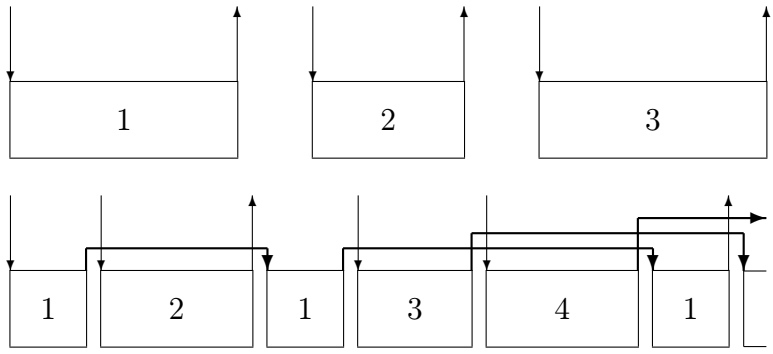
Skupina jazyků ALGOL: Původní jazyk ALGOL se změnil v jazyk ALGOL-68 a tím jeho vývoj skončil. Jeho syntaxi však převzal (a výrazně obohatil) jazyk Pascal z druhé poloviny 70. let. V tomto duchu pokračoval i jazyk MODULA-2 z konce let osmdesátých, ten se však velkého rozšíření nedočkal.

Novější jazyky: Další jazyky jsou novější – jazyk BASIC z roku 1964 (jenž se v 90. letech vyvinul v jazyk Visual BASIC), jazyk FORTH a zejména jazyk C z roku 1978 (v tomto roce byla vydána základní učebnice tohoto jazyka [1], dnešní oficiální standard ANSI C pochází z roku 1988) s jeho rozšířenou verzí C++. Vedle těchto programovacích jazyků byly učiněny aspoň dva pokusy o sjednocení – jazyk PL/I ze 70. let a jazyk ADA z 80. let.

Na fyzikálních pracovištích se nejčastěji setkáme s jazyky Pascal, C a C++, FORTRAN-77 a FORTRAN-90 a také s různými dialekty jazyka BASIC. V poslední době se začínají rozšiřovat i nové jazyky Delphi a JAVA. Z hlediska počítačové fyziky však mají výsadní postavení pouze dva jazyky – jazyk FORTRAN-90/95 a jazyk C. V těchto dvou jazycích je napsáno více než 90 % všech programů, přičemž jejich vzájemný poměr se blíží k 50 procentům (výrazně geograficky rozděleným – Japonsko téměř výhradně FORTRAN, USA převaha C, Evropa zhruba stejné zastoupení obou jazyků). Preference příslušného jazyka je též vlastností určité komunity – uživatelé operačního systému UNIX častěji preferují jazyk C, atd.

Vedle volby vhodného programovacího jazyka je druhou základní otázkou volba vhodného operačního systému. Co to je ***operační systém***? V počátcích počítačové fyziky byly všechny problémy řešeny ve strojovém kódu a veškerá obsluha počítače byla prováděna manuálně z operátorské konzole. S rozvojem výpočetní techniky se nejprve zefektivnila práce uživatele zavedením programovacích jazyků a v další generaci vývoje výpočetní techniky (Tab. 2.1) byla zefektivněna i práce obsluhy počítače zavedením operačního systému, který podobně jako programovací jazyk převzal mechanickou rutinní činnost, jež bylo možno předem naprogramovat. Tak vznikla celá řada operačních systémů, z nichž nejznámější jsou asi CP/M, MS-DOS, Windows a UNIX. Pro nás se volba redukuje – pokud budeme pracovat na mikropočítačích typu PC, je třeba rozhodnout mezi různými variantami operačního systému Microsoft Windows (98, 2000, XP) a systémem Linux. Při práci na pracovních stanicích a větší výpočetní technice se bude s největší pravděpodobností jednat o různé varianty operačního systému UNIX.

Další otázkou je volba ***režimu práce*** dané výpočetní techniky. Různé režimy práce si nejlépe objasníme graficky - viz obr. 2.2. Během vývoje výpo-



Obrázek 2.2: Režimy práce: individuální (nahoře) a multiprogramový (dole).

četní techniky a v závislosti na její výkonnosti a kapacitě paměti se používaly různé režimy, někdy i na jednom zařízení současně:

Individuální režim – Tento režim je nejstarší. Uživatel má při něm k dispozici počítač sám pro sebe. Když ukončí práci, přijde druhý uživatel a počne pracovat stejným způsobem (obr. 2.2 nahoře).

Dávkový režim – Pro urychlení práce jsou programy (a vstupní data) více uživatelů připraveny předem do vstupního zařízení počítače. Práci řídí operační systém, který ihned po dokončení výpočtu jednoho uživatele spustí výpočet dalšího a tak omezí prostoje znázorněné na obr. 2.2 nahoře mezerami mezi bloky.

Multiprogramování – I při dávkovém režimu se často stávalo, že počítač nebyl plně využit. Bylo to tehdy, když program vyžadoval práci s pomalou periférií (např. vstupní nebo výstupní jednotkou). Byl proto zaveden režim multiprogramovací, při kterém jsou v paměti počítače uloženy programy více uživatelů a tyto programy mají různé priority – na obr. 2.2 dole jsou označeny tak, že nejvyšší prioritu má program číslo 1, pak 2, atd. Pracovat začne program č. 1. Když uvolní procesor, aby po určitou dobu vykonával jiné operace, začne ihned pracovat další program. Jakmile však původní program vyžaduje opět procesor pro sebe, vzhledem k jeho vyšší prioritě ten další program přeruší práci a původní program pokračuje v činnosti. Tento postup se může vícekrát opakovat. Na obrázku jsou znázorněny různé situace, při kterých další program např. může ukončit úspěšně výpočet a výpočet zahájí třetí program, atd.

Existuje více druhů multiprogramování, které se liší v pravidlech, jak si programy předávají řízení. Vedle způsobu znázorněného na obr. 2.2 dole

je obvyklý i způsob, při kterém mají všechny programy stejnou prioritu a ve výpočtech se střídají cyklicky po určitých krátkých intervalech.

V experimentální fyzice je běžný způsob, při kterém je v počítači přítomen jeden program s maximální prioritou a další program nebo programy mají prioritu nízkou. Ten hlavní program obsluhuje experimentální aparaturu a okamžitě reaguje na její požadavky. Ve zbylém čase lze provádět výpočty, kreslit, psát diplomové práce, apod.

Obecně lze říci, že všechny způsoby multiprogramování výrazně zvyšují efektivitu práce celého systému. V příkladu uvedeném na obr. 2.2 dole program č. 1 provede svůj výpočet stejnou rychlostí jako v individuálním nebo dávkovém režimu, zatímco programy č. 2, 3, ... budou spočítány rychleji.

Na velkých počítačích (sálových systémech nebo superpočítačích) se v současné době pracuje pouze v režimu multiprogramování. Individuální režim byl provozován pouze v počátcích – v 50. letech. V 60. letech se většinou začalo přecházet na dávkový režim a 70. až 80. letech na multiprogramovací. Mikropočítače obecně kopírují vývoj velkých systémů s určitým skluzem. Proto osobní mikropočítače pracují většinou v individuálním režimu, ale výkonnější z nich a pracovní stanice pro zvýšení efektivity práce jsou též multiprogramovatelné – ať již systémem více uživatelů (terminálový provoz) nebo spouštěním více programů jednoho uživatele. Tento způsob práce podporují i současné operační systémy na bázi Windows nebo Linuxu.

2.3 Zásady programování

I když máme k dispozici moderní výpočetní techniku a překladač vhodného programovacího jazyka, je možné vytvořit programy nejrůznější kvality. Efektivním vytváření programů se sice zabývá jiná vědní disciplína – součást informatiky, softwarové inženýrství, základy vytváření dobrých programů však musí znát každý počítačový fyzik (a měl by je znát vůbec každý fyzik).

V počátcích byl jediným kritériem při vytváření programů požadavek na vznik programu, který správně vyřeší zadaný problém a to jakýmkoliv způsobem. Důsledkem často byly nepřehledné programy, neefektivně vytvářené, těžko čitelné, velmi těžko udržovatelné a rozšiřitelné. Této fázi se dnes často říká „*divoké programování*“ a vzniklým programům pejorativně „spaghetti code“. Není ji možno zcela odsuzovat, neboť touto technikou bylo dosaženo často velmi cenných výsledků.

V druhé polovině 70. let však vznikla myšlenka ***strukturovaného programování***, která se ukázala velmi vhodnou ve vědecké práci. Zásady struk-

turovaného programování jsou shrnuty v knize N. Wirtha z roku 1976 s typickým názvem „Algorithms + Data Structures = Programs“ [2]. Tuto techniku lze stručně charakterizovat jako zavedení dobrovolných omezení při vytváření programů s cílem zefektivnit tvorbu, čitelnost a udržování programů. Znamená to tedy nevyužívat všechny možnosti, které daný programovací jazyk poskytuje. To se týká především jazyka FORTRAN (a to varianty FORTRAN-77; novější varianty -90 a -95 již berou zásady strukturovaného programování v úvahu), který vznikl před zavedením metodiky strukturovaného programování a obsahuje příkazy, které jsou s ní neslučitelné – např. aritmetický podmíněný příkaz. Naopak na základě myšlenek strukturovaného programování vznikly jazyky, které strukturovanou práci přímo podporují – zde je třeba jmenovat především jazyk Pascal, autor N. Wirth (původní kniha prof. Wirtha [2] uvádí příklady v jazyku Pascal, její přepracované vydání z roku 1986 již pracuje s jazykem MODULA-2).

Uplynulých 25 let ukázalo, že dodržování zásad strukturovaného programování při práci v počítačové fyzice je velmi žádoucí. Dlouhodobá statistická analýza programů různé délky N (v počtu instrukcí) připravovaných jak pomocí divokého tak i strukturovaného programování ukázala, že dobu tvorby programu T můžeme popsat pomocí jednoduchého vztahu

$$T \simeq K \cdot N^m$$

kde exponent m nabývá zcela rozdílných hodnot pro obě programovací techniky: $m \doteq 1,1$ pro strukturované programování zatímco $m = 2 \div 3$ pro divoké programování. Je zcela zřejmé, že pro delší programy je technika divokého programování velmi nevhodná.

> Je ale nutno objektivně přiznat, že strukturované programování nepřináší pouze výhody. Pokud se testují kvalitně napsané programy strukturované a nestrukturované, ty druhé jsou typicky o 5 % výkonnější. Je to způsobené tím, že dobrovolné nevyužití všech možností daného programového jazyka znamená obvykle vytvoření trochu méně efektivního kódu. Avšak vzhledem k současné výkonnosti výpočetní techniky je značná úspora času programátora i dalších uživatelů považována za výrazně převažující výhodu.

Všechny zásady strukturovaného programování nepřežily zkoušku časem stejně dobře, uvedeme si proto jen ty základní, které by bylo skutečně velmi vhodné dodržovat:

1. *Programování top-down a bottom-up*
   Vytváření programu by mělo mít dvě fáze. Nejprve je nutno provést

analýzu problému bez použití počítače, tj. rozložit složitý problém na dílčí jednodušší problémy (bloky) již snadno řešitelné (směr „top-down“). Teprve pak přistoupit k vytváření vlastního programu – nejprve naprogramovat a odladit dílčí bloky a teprve z nich skládat (a opět postupně odlaďovat) větší celky a nakonec celý program (směr „bottom-up“).

2. *Přednostní používání jen určitých programových konstrukcí*
   Tato zásada říká, že se mají používat jen programové konstrukce, které nepovedou k vytváření nepřehledných programů. Nevhodné jsou příkazy typu nepodmíněný skok, aritmetický podmíněný skok (na rozdíl od logického podmíněného skoku) a naopak vhodné jsou příkazy vytvářející v programu přirozené bloky. Zde je vhodné se inspirovat jazykem Pascal, který byl právě k účelu strukturovaného programování vyvinut a využívat jeho typy konstrukcí.

3. *Modularita programu*
   Programovací jazyk by měl obsahovat konstrukce umožňující vytváření uzavřených struktur, které po odladění lze volat jen přesně definovaným způsobem a tak již nelze jejich obsah porušit. Jedná se o tyto konstrukce: podprogram (jazyk FORTRAN), blok (jazyk Pascal) a modul (jazyk MODULA-2). Moderní programovací jazyky obvykle obsahují 2-3 takové konstrukce současně (např. jazyk FORTRAN-90 obsahuje, i když s určitými omezeními, všechny tři). S tím souvisí i otázka lokálních a globálních proměnných spolu s mechanismem odstiňování proměnných.
   Modularita programu se chápe na dvou úrovních. Bloky vytvořené pomocí zásady 1 by měly být dostatečně malé (zásada KISS – "Keep It Small & Stupid"), aby je bylo možno plně pochopit a dobře odladit – typická je délka kolem 50 programových řádků. Tyto bloky by měly být vhodným způsobem „zapouzdřeny“ do samostatných procedur a pak by měly být složeny do větších celků již podle logiky řešeného fyzikálního problému.

Vedle těchto tří základních zásad strukturovaného programování je možno uvést ještě některé další. Program by měl být prostorově členěn – vodorovně i svisle – pro zlepšení jeho čitelnosti (příkladem je opět jazyk Pascal), měl by být dostatečně dokumentován (a to komentáři vloženými přímo do programu), atd.

Vývoj metodiky programování pokračoval i po zavedení strukturovaného programování. Jako další etapa je uváděno tzv. *objektově orientované programování*. Jeho techniky jsou implementovány do některých současných pro-

gramovacích jazyků, především jazyka C++. I když přináší řadu výhod, za nevýhodu v oblasti vědecko-technických výpočtů se považuje nižší (a to někdy i značně) efektivita výsledného kódu a proto se objektově orientované programování zatím v počítačové fyzice masově nerozšířilo.

2.4 Historický vývoj hardwarových a softwarových prostředků

S tím, jak se hardwarové a softwarové prostředky vyvíjely, je možno řešit metodami počítačové fyziky stále složitější problémy. Abychom si demonstrovali, jak možnosti počítačových fyziků s časem rostou, uvedeme si výsledky získané pomocí tzv. GAMMIX testu v posledních dvou desetiletích.

Tento test je jedním z řady testů, které byly navrženy pro testování výkonnosti výpočetní techniky v numerických výpočtech. Dalším byl např. test ERATOS, tj. Eratosthenovo síto pro určování prvočísel, pomocí něhož byly testovány celočíselné operace. Test GAMMIX má smíšenou povahu blízkou výpočtům ve fyzice, neobsahuje však výpočty funkcí typické pro operace v pohyblivé řádové čárce, jež jsou v některých oblastech fyziky (např. ve fyzice plazmatu) klíčové. V současnosti je již zastaralý, jeho cena je ale v nepřetržité řadě testů prováděných po dlouhé časové období. Proto jej nemohou plně nahradit ani v současné době používané výrazně objektivnější testy typu SiSoft Sandra 200x, PCMark 200x, apod.

Nejprve si uvedeme výpis programu v jazyku FORTRAN:

```
Program GAMMIX
   integer(4)  I,J,K,L,N
   real(4)     S,X
   real(4)     A(100),B(30)
   X=0.12345
   do I=1,100
      A(I)=I
   end do
   do I=1,30
      B(I)=I
   end do
   write(*,*) "Pocet opakovani testu: "
   read(*,*) N
   do K=1,N
      do I=1,1000
         S=A(11)
```

```
         do J=1,10
            L=11-J
            S=S*X+A(L)
         end do
      end do
      do I=1,33
         S=0.0
         do J=1,30
            S=S+A(J)*B(J)
         end do
      end do
      do I=1,667
         do J=1,30
            B(J)=A(J)+B(J)
         end do
      end do
      do I=1,1333
         S=X
         do J=1,5
            S=(S-X/S)*0.5
         end do
      end do
      do I=1,67
         S=A(1)
         do J=2,100
            if (A(J).ge.S) S=A(J)
         end do
      end do
   end do
   stop
end
```

Jak vidíme, test kombinuje operace často používané ve vědecko-technických výpočtech – Hornerovo schéma na výpočet polynomu, skalární součin dvou vektorů, Newtonovu metodu počítání druhé odmocniny, hledání maxima, atd.

Výsledky získané při použití testu GAMMIX na různé výpočetní technice a v různých programovacích jazycích jsou uvedeny v následujícím přehledu (Tabulka 2.2). Z tohoto přehledu je zřetelně vidět, jak výrazně se výkonnost výpočetní techniky zvýšila za posledních dvacet let – přehled byl vypracováván v letech 1980–2003:

Počítač, procesor	**Programovací jazyk**	**Čas [s]**
I8080, Z80,...	BASIC	$\sim 1\,000$
I8086 (4,77 MHz)	Turbo Pascal 3.0	45
ZX Spectrum (Z80 3,5 MHz)	HiSoft Pascal	25
ATARI ST (8 MHz)	FORTRAN-77	7,7
I80286 (16 MHz)	Turbo Pascal 6.0	2,6
ICL 4-72	FORTRAN-77	1,24
Pentium (75 MHz)	Famulus	0,26
I80386 (33 MHz)+koprocesor	Salford FORTRAN-77	0,19
IBM 4381	FORTRAN-77	0,099
I80486 (66 MHz)	Salford FORTRAN-90	0,033
	Microsoft FORTRAN 32, v.1.0	0,024
Hewlett-Packard 715/75	FORTRAN-77/UNIX	0,0153
Pentium (75 MHz)	Microsoft FORTRAN 32, v.1.0	0,0090
Pentium (166 MHz)	Microsoft FORTRAN 32, v.4.0	0,0044
Pentium Pro (200 MHz)	Microsoft FORTRAN 32, v.4.0	0,0025
Pentium II (300 MHz)	Microsoft FORTRAN 32, v.4.0	0,0023
Pentium (166 MHz)	Digital FORTRAN 6.0	0.00187
Digital Alpha (433 MHz)	Microsoft FORTRAN 32, v.5.0	0,00052
Pentium II (450 MHz)	Digital FORTRAN 6.0	0,00040
Digital Alpha (600 MHz)	FORTRAN/UNIX	0,00039
Pentium III (1 GHz)	Compaq FORTRAN 6.5	0,000175
	Compaq FORTRAN 6.6	0,000047
Pentium 4 (1,8 GHz)	Compaq FORTRAN 6.6	0,0000259
AMD Athlon XP (1.8+)	Compaq FORTRAN 6.6	0,0000258
AMD Athlon XP (2.4+)	Compaq FORTRAN 6.6	0,0000199
Pentium 4 (2,53 GHz)	Compaq FORTRAN 6.6	0,0000184
Pentium 4 (3,06 GHz)	Compaq FORTRAN 6.6	0,0000154

Tabulka 2.2: Výsledky GAMMIX testu.

Výsledky testu jsou seřazeny podle rychlosti – od nejpomalejších (osmibitová výpočetní technika používaná ve školství na počátku 80. let, mikropočítače PMD 80, IQ-151, ZX 81 a ZX Spectrum, interpretační programovací jazyk BASIC) až po současné výkonné mikropočítače.

Povšimněte si, že výsledky testu nezávisejí pouze na výkonnosti procesoru, ale do značné míry i na použitém programovacím jazyku. To se ale netýká relativního skoku ve výkonnosti u mikroprocesoru Pentium III/1 GHz při přechodu ze staršího na novější překladač. Tento výkonnostní skok má jinou příčinu – nové překladače jsou již tak „chytré“, že poznají, že v programu

se vícekrát požaduje stejný výsledek, takže si ušetří práci a místo opakovaného výpočtu poskytnou již spočítaná data. U jiných testů, kde se žádné části neopakují, se tak výrazný skok ve výkonnosti procesoru nevyskytuje.

Vedle kombinovaného GAMMIX testu byly po dlouhou dobu prováděny i další testy. Uveďme si proto vybrané výsledky i dalších dvou z nich: TEST1 (Tab. 2.3) a TEST2 (Tab. 2.4).

Počítač, procesor	**Programovací jazyk**	**Čas** [s]
ZX Spectrum (Z80 3,5 MHz)	BASIC	65 000
I80286 (16 MHz)	Turbo Pascal 6.0	62
I80486 (66 MHz)	Microsoft FORTRAN 32, v.1.0	1,11
Pentium (75 MHz)	Microsoft FORTRAN 32, v.1.0	0,50
Pentium II (300 MHz)	Microsoft FORTRAN 32, v.4.0	0,105
Digital Alpha (600 MHz)	FORTRAN/UNIX	0,060
Pentium II (450 MHz)	Digital FORTRAN 6.0	0,058
Pentium III (1 GHz)	Compaq FORTRAN 6.6	0,0235
Pentium 4 (1,8 GHz)	Compaq FORTRAN 6.6	0,0150
Pentium 4 (2,53 GHz)	Compaq FORTRAN 6.6	0,0105
AMD Athlon XP (2.4+)	Compaq FORTRAN 6.6	0,0092
Pentium 4 (3,06 GHz)	Compaq FORTRAN 6.6	0,0089

Tabulka 2.3: Výsledky testu TEST1.

Oba poslední testy vznikly úpravou reálných programů z počítačové fyziky – TEST1 obsahuje pouze celočíselné operace a je typický pro některé algoritmy v oblasti zpracování obrazu, TEST2 má těžiště v operacích s pohyblivou řádovou čárkou a odpovídá algoritmům částicového modelování metodou molekulární dynamiky.

Převážná většina výsledků byla získána na mikropočítačích běžných v příslušné době (s mikroprocesory Intel, Zilog, Motorola, ...), některé testy proběhly i na profesionálních pracovních stanicích (Hewlett-Packard, Digital Alpha) a na sálových počítačích. Britský počítač ICL 4-72 byl špičkovým počítačem českých vysokých škol v první polovině 80. let a počítač IBM 4381 byl ve školství využíván začátkem 90. let. Jejich výkonnost ve všech testech se však příliš nelišila od běžných mikropočítačů té doby (značné rozdíly, které samozřejmě existovaly, se týkaly spolehlivosti systémů, kapacity paměti, programového vybavení, periférií, atd., ne však prostého výpočetního výkonu). Z testů je zřejmé, že riscové procesory pracovních stanic jsou vhodnější pro

Počítač, procesor	**Programovací jazyk**	**Čas** [s]
ZX Spectrum (Z80 3,5 MHz)	BASIC	195
I80286 (16 MHz)	Turbo Pascal 6.0	0,28
I80486 (66 MHz)	Microsoft FORTRAN 32, v.1.0	0,0084
Pentium (75 MHz)	Microsoft FORTRAN 32, v.1.0	0,0036
Pentium II (300 MHz)	Microsoft FORTRAN 32, v.4.0	0,00084
Pentium II (450 MHz)	Digital FORTRAN 6.0	0,00059
Digital Alpha (600 MHz)	FORTRAN/UNIX	0,00023
Pentium III (1 GHz)	Compaq FORTRAN 6.6	0,000226
Pentium 4 (1,8 GHz)	Compaq FORTRAN 6.6	0,000101
Pentium 4 (2,53 GHz)	Compaq FORTRAN 6.6	0,000072
AMD Athlon XP (2.4+)	Compaq FORTRAN 6.6	0,000069
Pentium 4 (3,06 GHz)	Compaq FORTRAN 6.6	0,000060

Tabulka 2.4: Výsledky testu TEST2.

operace v pohyblivé řádové čárce (TEST2), zatímco při celočíselných operací se více projeví frekvence procesoru bez ohledu na jeho architekturu (viz TEST1) – rozdíly však nejsou příliš velké.

> Za profesionální programovací jazyky jsou v počítačové fyzice považovány dva - jazyk FORTRAN (v různých variantách) a jazyk C (obvykle však ne jazyk C++). Ve výše uvedeném přehledu nejsou výsledky získané s programovacím jazykem C uvedeny, obecně však lze prohlásit, že výsledky získané na příslušném počítači jsou při použití jazyka C typicky o 10–30 % horší než při použití jazyka FORTRAN.

2.5 Shrnutí

Tato kapitola měla za úkol poskytnout přehledné informace o prostředí, v němž budou různé fyzikální problémy řešeny postupy počítačové fyziky. Tyto postupy budou detailně popsány v dalších kapitolách tohoto i dalšího dílu studijního textu.

Z látky uvedené v této kapitole má největší váhu sekce *2.3 Zásady programování*, neboť popisuje techniky, které musí studenti používat při vypracování svých programů. Proto tyto znalosti budou vyžadovány i u zkoušek.

Kapitola 3

POČÍTAČOVÉ MODELOVÁNÍ

Mezi všemi základními směry počítačové fyziky má mimořádné postavení *počítačové modelování*. V literatuře se též vyskytují pojmy *počítačová simulace* a *počítačové experimenty*. Mezi těmito pojmy sice existuje drobný rozdíl, v prvním přiblížení však tím můžeme chápat totéž. Co to tedy je počítačové modelování?

3.1 Počítačové modelování

Při řešení fyzikálního problému počítačovým modelováním musíme projít následujícími kroky:

1. Formulace problému
   Při formulaci problému vyjdeme z fyzikálního jevu, který chceme studovat. Tento jev popíšeme pomocí pojmů, které budou odpovídat zvolené technice modelování (viz sekce 3.2).

2. Vytvoření modelu
   Na bázi studovaného jevu zformulujeme model, který budeme dále na počítači řešit. Vztah původního jevu a vytvořeného modelu je složitý:

   (a) Model je prakticky vždy jednodušší než studovaný jev
   Při formulaci modelu se dopouštíme zjednodušení – obvykle proto, že všechny vlastnosti studovaného jevu neznáme, někdy ale i proto, abychom byli schopni model na dostupné výpočetní technice vyřešit.

(b) Při formulaci modelu nemáme jistotu o jeho správnosti
Při vytváření modelu se snažíme do něj zahrnout základní rysy studovaného fyzikálního jevu a vynechat pouze méně podstatné vlastnosti, nevíme ale, zda se nám to plně podařilo a též nevíme, zda jsou vůbec ty podstatné rysy známy. Proto je nutno výsledky modelování srovnat s experimentálními daty – viz krok 4.

(c) Výsledky počítačového modelování vypovídají pouze o modelu
To, co počítačovým modelováním získáme, vypovídá pouze o zformulovaném modelu a jen nepřímo o studovaném jevu. Pokud jsme model příliš zjednodušili či dokonce zformulovali chybně, budou tyto chyby přítomné i ve výsledcích modelování. Zejména často se zapomíná na přesnost vstupních experimentálních dat, na nichž je model založen. I když nám počítač je schopen poskytnout výsledky na mnoho desetinných míst, skutečná přesnost výsledků modelování je dána přesností dat na vstupu – což jsou obvykle procenta až desetiny procent, v některých oblastech fyziky ale i desítky procent až celé řády.

3. Řešení modelu
Podle toho, jakou technikou budeme model řešit, zvolíme patřičné numerické metody. Obvykle se jedná o řešení soustav obyčejných nebo parciálních diferenciálních rovnic, případně rovnic integrodiferenciálních, někdy ale i postupy počtu pravděpodobnosti a matematické statistiky. Obecně lze říci, že v tomto bodě se opíráme o standardní metody různých oblastí matematiky.

4. Srovnání výsledků modelování s experimentálními údaji
Jak již bylo výše naznačeno, výsledkům modelování nelze z různých důvodů plně věřit. Abychom tuto nejistotu zmenšili, je třeba zabudovat do počítačového experimentu „zpětnou vazbu“. Tím se rozumí to, že výsledky modelování srovnáme s výsledky přímého měření. Zde máme dvě možnosti. Lze využít vstupní data, na nichž byl náš model postaven, a nebo nezávislé experimentální údaje. Použití vstupních dat modelu je sice možné, nic nám to však nevypovídá o správnosti zformulování modelu ale jen o tom, že jsme se při řešení modelu nedopustili chyb (i tato informace je cenná a takový test má smysl). Správný test modelu je ale založen na nezávislých datech, která nebyla při formulování modelu známá a která by měla vyjít z výsledku modelování sama. Pokud tomu tak nebude a model povede na odlišné výsledky (částečně nebo úplně), je třeba vstupní předpoklady modelu opravit, model znovu zformulovat a projít znovu celým cyklem kroků $2 \to 3 \to 4 \to \cdots$.

> Při počítačovém modelování je možné vědomě řešit i zcela chybný model. Jedná se o jakési vybudování „alternativního světa“, kde platí jiné fyzikální zákony a my se počítačovým experimentem snažíme vyšetřit, jaký důsledek to bude mít. Taková informace nám někdy dokáže pomoci lépe pochopit náš studovaný jev. Tato technika však není obvyklá a měla by se používat jen opatrně a výhradně jen zkušenými uživateli.

Z výše uvedených kroků počítačového modelování je těžiště práce v přechodu mezi krokem 1 a 2, tj. v naformulování správného modelu, který bude optimálně řešit náš fyzikální problém. Zde je skutečná tvůrčí činnost počítačového fyzika, vše ostatní je jakési „řemeslo“, kterému je třeba se naučit.

3.2 Techniky počítačového modelování

Model může být formulován na různých úrovních zobecnění a tomu odpovídají i různé metody počítačového modelování:

- *Částicové techniky*
  Nejsilnějším prostředkem je popsat studovaný jev na mikroskopické úrovni, tj. detailně studovat chování částí, ze kterých se skládá. Slovo „mikroskopický“ se musí brát relativně – pevnou látku nebo plazma budeme popisovat pomocí chování jednotlivých atomů, iontů nebo elektronů, zatímco pro popis galaxie stačí pracovat s hvězdami. Při testování chování modelu (krok 4 předchozí sekce) obvykle musíme provést statistické vyhodnocení těchto dat, protože experimentální fyzik nám určitě není schopen poskytnou informaci např. o poloze a rychlosti všech atomů nebo molekul ve studovaném souboru (a u elektronů tomu navíc brání i relace neurčitosti), čímž část informace z modelování ztratíme. Tato informace nám však zůstane k dispozici a můžeme se k ní vracet a získávat ještě další výsledky.

  Podle toho, jak chování částí studovaného celku popisujeme, rozeznáváme několik metod:

  - Metoda Monte Carlo
    Zde chování jednotlivých částic popisujeme stochasticky pomocí zákonů počtu pravděpodobnosti.
  - Metoda molekulární dynamiky
    Jedná se o deterministickou metodu, kde numericky řešíme pohybové rovnice popisující studovaný soubor částic. Z výpočetního

hlediska se bude jednat o řešení souboru obyčejných diferenciálních rovnic.

– Hybridní techniky
Při těchto postupech kombinujeme na částicové úrovni deterministický a stochastický popis tak, aby výpočet byl co nejefektivnější.

- *Spojité modelování*
Druhým extrémem je popis studovaného jevu na makroskopické úrovni – např. na plazma se díváme jako na kontinuum o určité teplotě, tlaku, složení, rychlosti proudění, atd. Mezi těmito makroskopickými veličinami platí zákony zachování – energie, hybnosti, náboje, rovnice kontinuity, apod. Matematicky jsou tyto zákony zachování popsány parciálními diferenciálními rovnicemi, pouze při užití určitých aproximací můžeme někdy přejít na obyčejné diferenciální rovnice.

- *Hybridní modelování*
I na této úrovni můžeme použít techniky hybridního modelování, kdy kombinujeme přístup spojitý a částicový, opět s cílem získat výsledek co nejrychleji a s co největší přesností.

Pokud spojitý a částicový přístup navzájem porovnáme, tak lze konstatovat, že částicové metody přinášejí výsledky podstatně přesnější, ale též s výrazně větší spotřebou strojového času. Abychom náš jev popsali co nejúplněji, musíme pracovat současně s velkým počtem částic – typicky 10^3 až 10^6, někdy i více. Tomu odpovídá i velký počet rovnic, které musíme vyřešit. Naproti tomu při spojitém přístupu zformulujeme několik (obvykle 5 až 10 a jen výjimečně více) rovnic, jež jsou sice složitější než u částicového přístupu, ale přesto bývá výpočet mnohem kratší. Základní nevýhodou spojitého přístupu však je, že přináší obvykle jen hrubé kvalitativní výsledky. Právě proto se navrhují hybridní přístupy s cílem zachovat rychlost výpočtu spojitého přístupu a přesnost zvýšit částicovou technikou.

Výše uvedené dělení metod na mikroskopické (částicové) a makroskopické (spojité) představuje pouze dva extrémní případy z celého spektra metod. Vyjdeme-li z deterministického částicového přístupu, kdy známe v každém časovém okamžiku polohy a rychlosti všech částic studovaného souboru, tak různé zákony zachování dostaneme váženou integrací přes celý fázový prostor, tj. obvykle se jedná o dvojitý integrál přes polohy a rychlosti částic. Pokud se provede jen jedna integrace, dostaneme metody ležící „mezi“ těmi krajními případy. Z těchto metod je asi nejznámější použití *Boltzmannovy kinetické rovnice*, kde studovanou veličinou je rozdělovací funkce $f(\vec{r}, \vec{v}, t)$ závislá

obecně na poloze, rychlosti a čase. Boltzmannova kinetická rovnice v závislosti na své pravé straně, tj. na formulaci tzv. srážkového členu, může být parciální diferenciální rovnicí v šesti proměnných a nebo dokonce rovnicí integrodiferenciální.

Další otázkou je, v jakém prostředí náš model formulujeme – buď na úrovni klasické fyziky a nebo se započtením kvantové mechaniky. Je jasné, že kvantově-mechanický přístup bude podstatně složitější, používat jej však musíme jen při studiu jevů z mikrosvěta. Např. problémy z oblasti fyziky plazmatu (a samozřejmě tím spíše z astronomie) budeme studovat na úrovni klasické, zatímco přesný popis chování elektronů v polovodičovém prvku si již vyžádá přístup kvantově-mechanický.

3.3 Shrnutí

Nejpodstatnějším pojmem zavedeným v této kapitole a pojmem základním pro celou počítačovou fyziku je pojem *model* a jeho vztah k realitě.

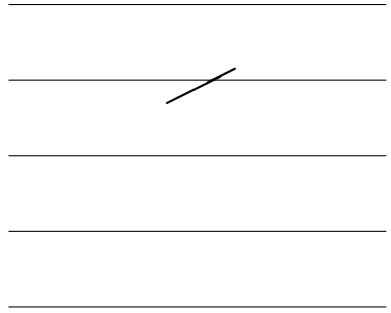
Kapitola 4

METODA MONTE CARLO

4.1 Princip metody

Metodou Monte Carlo se nazývá souhrn postupů dovolujících pomocí mnohonásobných náhodných pokusů získat řešení problémů z nejrůznějších oblastí vědy, tato metoda nepatří tedy pouze počítačové fyzice. Všeobecně se předpokládá, že metoda vznikla současně s rozšířením počítačů. V roce 1944 navrhl von Neumann použít aparát teorie pravděpodobnosti za pomoci počítače při pracích na atomovém výzkumu. Mezi tvůrce metody patří J. von Neumann, S. A. Ulam, N. Metropolis, H. Kahn a E. Fermi. Termín „Monte Carlo“ byl poprvé použit v publikaci Metropolise a Ulama z roku 1949 [3].

Princip metody je však mnohem starší, neboť k realizaci náhodných pokusů počítač vůbec nepotřebujeme. Pravděpodobně nejstarší aplikací metody Monte Carlo (i když se tak ještě nejmenovala) je tzv. Buffonova úloha o jehle. Jedná se o empirickou metodu na určení hodnoty čísla π. Metoda byla po-



Obrázek 4.1: Buffonova jehla.

psána a matematicky analyzována v článku Halla [4] z roku 1873. Princip metody je však ještě o sto let starší – objev patří Buffonovi [5] (Georges Louis Leclerc, Comte de Buffon) a to již v roce 1777.

Popišme si princip metody (obr. 4.1). Na horizontální rovině nakresleme rovnoběžky ve vzdálenosti d od sebe a mějme jehlu o délce $l \leq d$. Náhodný pokus budeme realizovat tak, že jehlu náhodně hodíme na rovinu. Výsledek pokusu bude úspěšný tehdy, když jehla protne jednu z čar, jinak bude neúspěšný. Pokus N-krát zopakujeme a budeme určovat číslo M/N, kde M je počet úspěšných pokusů. Jednoduchým výpočtem se dá najít limitní vztah mezi poměrem M/N pro nekonečný počet pokusů a délkami l a d. V konstantě úměrnosti v tomto vztahu je obsaženo číslo π, takže nalezením poměru M/N experimentálně určíme hodnotu konstanty π. Toto zjištění bylo ve své době velmi překvapující, neboť náhodným procesem bylo možno určit univerzální konstantu a to teoreticky s libovolnou přesností. Prakticky však je při konečném počtu hodů výsledek zatížen značnou chybou: při 10 pokusech jsme schopni odhadnout hodnotu π jako 3, při 1 000 pokusech jako 3,1, při 100 000 pokusech jako 3,14, atd.

Buffonova jehla proto asi nebude tou nejefektivnější cestou pro určování přesné hodnoty čísla π, ale je to velmi názorný příklad, jak lze řešit (téměř libovolný) problém pomocí metody Monte Carlo.

Základní schéma řešení problému pomocí metody Monte Carlo se bude skládat z těchto etap:

1. Analýza problému a vytvoření modelu
   Tento bod odpovídá počátečním dvěma krokům obecné struktury modelování (Sekce 3.1). Jelikož se bude jednat o stochastickou metodu modelování, cílem analýzy zkoumaného jevu bude jej popsat pomocí náhodné veličiny. Ve fyzice je tento požadavek dosti často přirozený, neboť řada studovaných fyzikálních problémů je svou podstatou náhodná. Jak jsme ale viděli na případu Buffonovy jehly, lze i zcela nenáhodný problém formulovat stochasticky.

   Vytvoření modelu pak znamená zjednodušeně popsat zkoumaný jev pomocí konkrétní náhodné veličiny s daným oborem hodnot a rozdělením pravděpodobností (viz Sekce 4.2) a současně určit, která charakteristika této náhodné veličiny obsahuje námi hledanou odpověď.

2. Generování náhodné veličiny
   Při počítačové simulaci metodou Monte Carlo musíme náhodnou veličinu vytvořit na počítači. To se obvykle provádí ve dvou krocích. Nejprve nagenerujeme na počítači náhodnou veličinu s určitým pevně daným rozdělením pravděpodobnosti - tento krok.

3. Transformace náhodné veličiny
   Potom podle požadavků našeho modelu tuto veličinu přetransformujeme v hledanou náhodnou veličinu. Tento postup je mnohem jednodušší než vytvářet přímo generátory nejrůznějších náhodných veličin.

4. Opakování kroků 2 a 3 a statistické vyhodnocení výsledků
   Jelikož se jedná o stochastickou metodu modelování, předchozími kroky dostaneme pouze jednu realizaci hledané náhodné veličiny. Proces proto musíme opakovat a protože metoda Monte Carlo konverguje pomalu, počet opakování musí být veliký. Tímto postupem dostaneme potřebný počet konkrétních realizací naší náhodné veličiny $\xi_1, \xi_2, \ldots, \xi_N$. Tyto hodnoty podrobíme statistické analýze a podle formulace našeho modelu z nich získáme hledanou odpověď.

V případě experimentálních realizací stochastického modelování (které se občas ještě provádí, i když význam počítačových realizací metody Monte Carlo stále narůstá), kroky 2 a 3 spolu splývají.

Ještě si povšimněme chyby metody Monte Carlo. Empiricky při pokusech s Buffonovou jehlou (stejně jako při dalších počítačových i experimentálních stochastických simulacích) bylo nalezeno, že chyba metody klesá s počtem pokusu N podle vztahu

$$\vartheta \sim 1/\sqrt{N}\,. \tag{4.1}$$

Jak jsme již obecně uvedli v předchozí kapitole, počítačový experiment pomocí metody Monte Carlo se skládá z části tvůrčí, kterou je krok 1, a rutinních částí, jež představují kroky 2 až 4. Tyto rutinní části si nyní podrobněji probereme, dříve však se musíme seznámit se základními pojmy z teorie pravděpodobnosti a matematické statistiky, neboť na nich je celá metoda Monte Carlo založena.

4.2 Základy počtu pravděpodobnosti a matematické statistiky

V této sekci bude uveden velmi stručný přehled základních pojmů z počtu pravděpodobnosti spolu s vybranými pravidly, jak s těmito pojmy pracovat. Výběr témat je podřízen cíli – metodě Monte Carlo – a zdaleka není vyčerpávající. Existuje však velké množství literatury (a to i české), kde lze získat další informace – např. [6], [7], atd.

4.2.1 Zavedení náhodných veličin

V praxi se často setkáváme s veličinami, u kterých známe všechny hodnoty, kterých mohou nabývat, a pravděpodobnosti jejich nabytí, u nichž však nedovedeme předpovědět jejich hodnotu v konkrétním případě. Takovým veličinám budeme říkat ***náhodné veličiny***. V tomto studijním textu budeme náhodné veličiny označovat písmeny řecké abecedy. Klasickým příkladem náhodné veličiny je výsledek házení hrací kostkou.

Náhodné veličiny dělíme na *diskrétní* a *spojité* podle toho, jakých hodnot mohou při realizacích nabývat. Diskrétní náhodná veličina může nabývat spočetně (konečně nebo nekonečně) mnoha hodnot, možnými hodnotami spojitých náhodných veličin jsou všechna čísla z konečného nebo nekonečného intervalu.

Pro každou náhodnou veličinu se zavádí tzv. „zákon rozdělení náhodné veličiny“, který každé hodnotě (nebo množině hodnot z určitého intervalu) přiřazuje pravděpodobnost, že náhodná veličina této hodnoty nabude. Existují dvě varianty formulace tohoto zákona. Pomocí

Distribuční funkce: Distribuční funkce přiřazuje každému reálnému číslu pravděpodobnost, že náhodná velikost nabude hodnot menších než toto číslo.Vezměme náhodnou veličinu ξ a její distribuční funkci označíme $F(x)$. Tuto funkci zavedeme předpisem

$$F(x) = P\{\xi < x\}\,, \tag{4.2}$$

kde symbolem $P\{y\}$ se označuje pravděpodobnost výskytu jevu y. Pravděpodobnost nemožného jevu je nulová a pravděpodobnost jistého jevu je rovna jedné. Distribuční funkce $F(x)$ proto nabývá hodnot z intervalu $\langle 0, 1\rangle$, je neklesající a platí pro ni

$$P\{x_1 \leq \xi < x_2\} = F(x_2) - F(x_1)\,,$$

pro $x_1 < x_2$.

Pravděpodobnosti: Distribuční funkce má díky své definici (4.2) integrální význam, shrnuje výslednou pravděpodobnost za určitý interval. Pokud chceme pracovat s konkrétními pravděpodobnostmi, musíme zavést odpovídající diferenciální veličiny, např. $p(x)$

$$p(x) = \frac{\mathrm{d}F(x)}{\mathrm{d}x}\,,$$

neboli

$$F(x) = \int_{-\infty}^{x} p(y)\,\mathrm{d}y\,.$$

Oba popisy jsou ekvivalentní, ale v počítačové fyzice dáváme častěji přednost práci s pravděpodobnostmi, neboť jsou bližší fyzikálnímu způsobu vyjadřování.

Nyní si tyto údaje upřesníme pro oba typy náhodných veličin – diskrétní a spojité.

1. Diskrétní náhodné veličiny
   U diskrétních náhodných veličin můžeme zákon rozdělení náhodné veličiny popsat množinou hodnot x_i a odpovídajícími pravděpodobnostmi p_i, kde $p_i = P\{\xi = x_i\}$

$$\xi = \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ p_1 & p_2 & \dots & p_n \end{pmatrix}. \tag{4.3}$$

   Hodnoty $x_1, \dots, x_n$, kterých může náhodná veličina ξ nabývat, mohou být libovolná čísla. Pro pravděpodobnosti $p_1, \dots, p_n$ však máme dvě omezení

$$p_i > 0, \quad i = 1, 2, \dots, n$$

$$\sum_{i=1}^{n} p_i = 1\,.$$

   Místo pravděpodobností p_i můžeme samozřejmě používat i distribuční funkci $F(x)$, která bude mít v tomto případě tvar skokové funkce.

2. Spojité náhodné veličiny
   Spojitá náhodná veličina ξ nabývá hodnot x z nějakého konečného nebo nekonečného intervalu. Pro zápis pravděpodobnostní informace můžeme použít jak distribuční funkci $F(x)$, která bude pro spojitou náhodnou veličinu spojitá, tak tzv. „hustotu pravděpodobnosti náhodné veličiny ξ v bodě x“, $p(x)$. Úplná charakteristika spojité náhodné veličiny bude proto mít tvar

$$\xi: \quad x \in \langle a, b\rangle, \quad p(x)\,. \tag{4.4}$$

   Hustota pravděpodobnosti $p(x)$ má tyto vlastnosti:

$$p(x) \geq 0, \quad x \in \langle a, b\rangle$$

$$\int_{-\infty}^{\infty} p(x)\,\mathrm{d}x = 1$$

$$P\{x_1 \leq \xi < x_2\} = \int_{x_1}^{x_2} p(y)\,\mathrm{d}y\,.$$

S náhodnými veličinami můžeme provádět různé matematické operace:

– Budou-li ξ a η nezávislé náhodné veličiny a c číselná konstanta, budou např. i $\xi + \eta$, $\xi + c$, $c \cdot \xi$ a $\xi \cdot \eta$ též náhodné veličiny.

– Spojité náhodné veličiny mohou být i argumenty běžné matematické funkce $f(x)$ a výsledek bude novou náhodnou veličinou $\eta = f(\xi)$, kterou budeme nazývat náhodná funkce.

– Můžeme pracovat i s vícerozměrnými náhodnými veličinami. V tomto případě pro popis pravděpodobností můžeme použít sdruženou distribuční funkci, která má např. ve dvou rozměrech tvar

$$F(x, y) = P\{\xi < x, \eta < y\}\,.$$

4.2.2 Charakteristiky náhodných veličin

Informace uvedená ve vztazích (4.3) nebo (4.4) je úplná a zcela popisuje danou náhodnou veličinu. Někdy je však třeba použít zkrácenou formu zápisu udávající pouze základní vlastnosti náhodné veličiny. Důvodů může být více – např. to, že náhodnou veličinu vytváříme na základě experimentálních dat a jejich počet nestačí na úplnou rekonstrukci, nebo že použitá data jsou zatížena šumem, který vlastně vytváří pozorovaný náhodný proces a je proto zbytečné tento šum úplně popisovat.

Pro tento účel byl navržen systém charakteristik, neboli ***momentů náhodných veličin***. Každý z momentů je prosté číslo, které v sobě obsahuje část informace o náhodné veličině. Úplná informace by byla obsažena v nekonečné soustavě momentů, ale my se omezíme na popis pomocí jejich konečného počtu. Jelikož momenty tvoří hierarchickou posloupnost od významnějších k méně významným, odříznutí vyšších členů této posloupnosti automaticky redukuje zápis náhodné veličiny na popis jejích nejpodstatnějších rysů. Jelikož se jedná o hierarchickou posloupnost, nelze nikdy pracovat s vyššími momenty a nižší momenty přeskočit!

V historii matematiky existoval větší počet takových posloupností momentů, část z nich již zanikla, a my si uvedeme tu nejužívanější.

1. Charakteristiky polohy
   Prvním momentem a základní charakteristikou každé náhodné veličiny ξ je její *střední hodnota s ohledem na rozdělení pravděpodobností*. Označujeme ji $E\xi$ a nazýváme očekávaná hodnota, matematická naděje nebo matematické očekávání.

Definiční vztahy pro diskrétní a spojitou náhodnou veličinu jsou

$$\begin{aligned} E\xi &= \sum_i x_i \cdot p_i \\ E\xi &= \int_{-\infty}^{\infty} x \cdot p(x)\,\mathrm{d}x\,. \end{aligned} \tag{4.5}$$

Označíme-li symboly ξ a η náhodné veličiny a symbolem c číselnou konstantu, bude platit

$$\begin{aligned} Ec &= c \\ E(\xi + c) &= E\xi + c \\ E(c \cdot \xi) &= c \cdot E\xi \\ E(\xi + \eta) &= E\xi + E\eta \\ E(\xi \cdot \eta) &= E\xi \cdot E\eta, \end{aligned}$$

přičemž poslední vztah bude platit jen tehdy, budou-li veličiny ξ a η nezávislé.

To nám dává jednoduché a často používané kritérium na zjišťování nezávislosti dvou náhodných veličin a dokonce na kvantifikování stupně jejich případné závislosti.

Alternativní charakteristikou popisující polohu rozdělení veličiny ξ je *medián* $\tilde{x}$ definovaný současnými vztahy

$$\begin{aligned} P\{\xi \le \tilde{x}\} &\ge 1/2 \\ P\{\xi \ge \tilde{x}\} &\ge 1/2\,. \end{aligned}$$

Tento moment je prvním momentem anglosaské soustavy charakteristik náhodných veličin.

2. Charakteristiky variability
Další moment udává rozptyl možných hodnot náhodné veličiny ξ kolem její střední hodnoty $E\xi$. Definiční vztah (nyní již společný pro diskrétní i spojité náhodné veličiny, neboť rozdíly jsou obsažené již v definici prvního momentu) pro druhý moment zvaný *rozptyl*, *variance* nebo *disperze* a označovaný $D\xi$, je

$$D\xi = E(\xi - E\xi)^2. \tag{4.6}$$

Tento vztah je nepraktický pro výpočet, proto se převádí na výhodnější ekvivalentní vyjádření

$$D\xi = E(\xi^2) - (E\xi)^2. \tag{4.7}$$

Označíme-li opět symboly ξ a η náhodné veličiny a c číselnou konstantu, bude pro rozptyl platit

$$\begin{aligned} Dc &= 0 \\ D(\xi + c) &= D\xi \\ D(c \cdot \xi) &= c^2 \cdot D\xi \end{aligned}$$

a pro nezávislé náhodné veličiny ξ a η bude navíc platit i

$$\begin{aligned} D(\xi + \eta) &= D\xi + D\eta \\ D(\xi - \eta) &= D\xi + D\eta\,. \end{aligned}$$

O rozptylu součinu dvou náhodných veličin $D(\xi \cdot \eta)$ nemůžeme na obecné úrovni prohlásit nic.

Vedle rozptylu $D\xi$ se zavádí odvozená jednotka zvaná *směrodatná odchylka*, $\sigma\xi$, pro níž platí

$$\sigma\xi = \sqrt{D\xi}\,.$$

3. Charakteristiky vyšších řádů
   Uvedený postup na získávání čísel, která charakterizují rozdělení náhodné veličiny ξ, můžeme zobecnit. Za předpokladu, že existuje $E\xi$ a má konečnou hodnotu, budeme definovat *centrální moment k-tého řádu*

   $$\mu_k = E(\xi - E\xi)^k, \quad k = 0, 1, 2, \ldots\,.$$

   Pro $k = 2$ dostaneme skutečně $\mu_2 = D\xi$.

   Momenty vyšších řádů mají ale jednu nepříjemnou vlastnost – vlivem mocniny v definičním vztahu jejich hodnota buď silně narůstá nebo klesá (v závislosti na vztahu momentu prvního řádu k hodnotě 1). Proto pro praktické účely bývá zvykem tyto momenty normalizovat:

   – Na základě momentu třetího řádu μ_3 je definována *šikmost*

   $$\alpha_3 = \frac{\mu_3}{\sigma^3}\,. \tag{4.8}$$

   U symetrických rozdělení je tato charakteristika nulová. Je-li kladná, je rozdělení pravděpodobnosti zešikmené doleva, je-li záporná tak doprava.

   – Normováním centrálního momentu 4. řádu definujeme *špičatost*

   $$\alpha_4 = \frac{\mu_4}{\sigma^4}\,. \tag{4.9}$$

Tato charakteristika nabývá pro Gaussovo rozdělení hodnotu 3. Bude-li $\alpha_4 > 3$, znamená to, že studované rozdělení je špičatější než rozdělení normální, pro menší hodnoty je rozdělení plošší. Zavádí se i výraz $\alpha_4 - 3$, jenž se nazývá *exces*.
Při konkrétním výpočtu se před definičními vztahy pro momenty vyšších řádů dává přednost vzorcům

$$\begin{aligned} \mu_3 &= E\xi^3 - 3 \cdot E\xi^2 \cdot E\xi + 2 \cdot (E\xi)^3 \\ \mu_4 &= E\xi^4 - 4 \cdot E\xi^3 \cdot E\xi + 6 \cdot E\xi^2 \cdot (E\xi)^2 + 3 \cdot (E\xi)^4. \end{aligned}$$

4.2.3 Vybrané náhodné veličiny

Nyní si uvedeme příklady náhodných veličin, s kterými se ve fyzikální praxi můžeme nejčastěji setkat. Nejprve si uvedeme vybrané *diskrétní* náhodné veličiny:

Binomické rozdělení

Tímto rozdělením se řídí četnost nějakého jevu v n nezávislých pokusech, když v každém pokusu má výskyt jevu pravděpodobnost p, kde n je přirozené číslo a $p \in (0, 1)$

$$\xi = \begin{pmatrix} 0 & 1 & \dots & n \\ p_0 & p_1 & \dots & p_n \end{pmatrix}, \tag{4.10}$$

$$p_x = P\{\xi = x\} = \begin{pmatrix} n \\ x \end{pmatrix} \cdot p^x \cdot (1-p)^{n-x}.$$

Základní charakteristiky rozdělení jsou

$$\begin{aligned} E\xi &= n\,p \\ D\xi &= n\,p\,(1-p) \\ \alpha_3 &= \frac{1-2p}{\sqrt{n\,p\,(1-p)}} \\ \alpha_4 &= \frac{1-6p\,(1-p)}{n\,p\,(1-p)} + 3\,. \end{aligned}$$

Poissonovo rozdělení

Toto rozdělení popisuje proces, při kterém studujeme četnost nějakého jevu v mnoha pokusech, když výskyt tohoto jevu v jednotlivém pokusu je jen velmi málo pravděpodobný. Z binomického rozdělení se získá limitním přechodem:

$p \to 0$, $n \to \infty$ při současné platnosti vztahu $n \cdot p = \lambda$ konečné. Toto λ je základním a jediným parametrem Poissonova rozdělení

$$\xi = \begin{pmatrix} 0 & 1 & \dots & n \\ p_0 & p_1 & \dots & p_n \end{pmatrix}, \tag{4.11}$$

kde

$$p_x = P\{\xi = x\} = e^{-\lambda} \cdot \frac{\lambda^x}{x!}\,.$$

Charakteristiky Poissonova rozdělení jsou

$$\begin{aligned} E\xi &= \lambda \\ D\xi &= \lambda \\ \alpha_3 &= \frac{1}{\sqrt{\lambda}} \\ \alpha_4 &= \frac{1}{\lambda} + 3\,. \end{aligned}$$

Platí tedy, že první dva momenty Poissonova rozdělení se přímo rovnají jeho parametru λ.

Součet dvou nezávislých Poissonových náhodných veličin s parametry λ_1 a λ_2 je opět Poissonovou náhodnou veličinou s parametrem λ rovným součtu dílčích parametrů $\lambda_1 + \lambda_2$.

Přesný vztah pro pravděpodobnosti p_x Poissonova rozdělení ve vzorci (4.11) je

$$p_x = \frac{1}{e^{\lambda} + 1} \cdot \frac{\lambda^x}{x!}\,.$$

Vztah uvedený v textu výše, jenž je v literatuře běžně používán, je zjednodušením platným pro větší hodnoty λ.

Rovnoměrné rozdělení

Diskrétní náhodná veličina ξ s rovnoměrným rozdělením může nabývat m hodnot $1, 2, \dots, m$ se stejnými pravděpodobnostmi

$$p = P\{\xi = x\} = \frac{1}{m}\,. \tag{4.12}$$

První dva momenty této veličiny jsou

$$\begin{aligned} E\xi &= \frac{m+1}{2} \\ D\xi &= \frac{m^2 - 1}{12}\,. \end{aligned}$$

A nyní si uvedeme příklady *spojitých* náhodných veličin. Se spojitými náhodnými veličinami se ve fyzice setkáváme častěji.

Rovnoměrné rozdělení

Spojitá náhodná veličina ξ zavedená v intervalu $\langle a, b\rangle$ má rovnoměrné rozdělení tehdy, má-li v tomto intervalu konstantní hustotu pravděpodobnosti

$$p(x) = \frac{1}{b-a} \quad x \in \langle a, b\rangle. \tag{4.13}$$

Základní charakteristiky rovnoměrného rozdělení jsou

$$\begin{aligned} E\xi &= \frac{a+b}{2} \\ D\xi &= \frac{(b-a)^2}{12} \\ \alpha_3 &= 0 \\ \alpha_4 &= \frac{9}{5}\,. \end{aligned}$$

Rovnoměrné rozdělení bývá při řešení problémů metodou Monte Carlo na počítači zvláště důležité, neboť náhodné veličiny popisující studovaný jev se obvykle získávají vhodnou transformací rovnoměrně rozdělené náhodné veličiny definované na intervalu $\langle 0, 1\rangle$. Tuto veličinu budeme ve studijním textu označovat řeckým písmenem gama, γ.

Gaussovo rozdělení

Gaussovo (neboli normální) rozdělení je nejdůležitějším rozdělením spojité náhodné veličiny a má základní význam v matematické statistice i v jejích aplikacích, stejně jako ve fyzice. Tímto rozdělením je možno popisovat celou řadu jevů. Mnohé náhodné veličiny, výsledky měření, jsou aspoň přibližně normální. Obecně bývá normální rozdělení pro popis daného jevu použitelné v těch případech, kdy na rozptyl hodnot náhodné veličiny působí současně velký počet nepatrných a navzájem nezávislých vlivů – zdůvodnění uvidíme v další sekci.

Gaussovo rozdělení je spojitou náhodnou veličinou s parametry

$$p(x) = \frac{1}{\sqrt{2\pi}\,\sigma} \cdot \exp\left[-\frac{(x-\mu)^2}{2\,\sigma^2}\right] \qquad x \in (-\infty, \infty)\,. \tag{4.14}$$

Jeho distribuční funkce je

$$p(x) = \frac{1}{\sqrt{2\pi}\,\sigma} \cdot \int_{-\infty}^{x} e^{-\frac{(t-\mu)^2}{2\,\sigma^2}}\,\mathrm{d}t\,.$$

Gaussovo rozdělení má dva parametry – μ a σ, kde $\sigma > 0$. Základní charakteristiky Gaussova rozdělení mají hodnoty

$$\begin{aligned} E\xi &= \mu \\ D\xi &= \sigma^2 \\ \alpha_3 &= 0 \\ \alpha_4 &= 3\,, \end{aligned}$$

neboli parametry Gaussova rozdělení jsou střední hodnotou a odmocninou z rozptylu.

Gaussovu náhodnou veličinu obvykle označujeme písmenem ζ nebo také $N(\mu, \sigma)$ – jako normální rozdělení. Speciálním případem je veličina $N(0, 1)$, která bývá tabelována.

Součet dvou nezávislých Gaussových náhodných veličin ζ_1 a ζ_2 je opět Gaussovou náhodnou veličinou.

> Z exponenciálního průběhu hustoty pravděpodobnosti normálního rozdělení plyne často používané *pravidlo tří sigma*, které tvrdí „Při jednom pokusu je prakticky nemožné získat hodnotu ζ lišící se od střední hodnoty o více než 3σ“. Zdůvodněním tohoto starého empirického pravidla je, že pro Gaussovu náhodnou veličinu ζ platí
>
> $$P\{\mu - 3\,\sigma < \zeta < \mu + 3\,\sigma\} = 0,9973$$
>
> a tato hodnota je velmi blízká jedné.

Maxwellovo rozdělení

Toto rozdělení, používané zejména v kinetické teorii plynů, má parametry

$$p(x) = \frac{2}{a^3\,\sqrt{2\pi}} \cdot x^2 \cdot \exp\left(-\frac{x^2}{2\,a^2}\right) \qquad x \in \langle 0, \infty) \,. \tag{4.15}$$

Rozdělení má jeden parametr $a > 0$. Základní charakteristiky rozdělení jsou

$$\begin{aligned} E\xi &= \frac{3\,a}{\sqrt{2}} \\ D\xi &= 3\,a^2\,. \end{aligned}$$

4.2.4 Vybrané limitní věty

Z velkého množství vět teorie pravděpodobnosti si uvedeme ty, na nichž je založeno statistické vyhodnocování experimentálních dat a modelování reálných procesů metodou Monte Carlo.

Zákon velkých čísel

Budou-li $\xi_1, \xi_2, \ldots, \xi_n$ nezávislé náhodné veličiny se stejným rozdělením, tj. se stejnými středními hodnotami μ, pak jejich aritmetický průměr

$$\bar{\xi} = \frac{1}{n} \cdot \sum_{i=1}^{n} \xi_i \tag{4.16}$$

konverguje podle pravděpodobnosti k μ. Toto tvrzení znamená, že pro každé kladné ε platí

$$\lim_{n\to\infty} P\left\{ \left| \frac{1}{n} \cdot \sum_{i=1}^{n} \xi_i - \mu \right| < \varepsilon \right\} = 1\,.$$

Pro větší počet nezávislých pozorování náhodné veličiny ξ můžeme proto jejich aritmetický průměr použít pro odhad střední hodnoty $E\xi$.

S využitím vlastností rozptylu náhodné veličiny ξ můžeme určit i rozptyl aritmetického průměru $\bar{\xi}$

$$D\bar{\xi} = \frac{D\xi_1 + \ldots + D\xi_n}{n^2} = \frac{\sigma^2}{n}\,. \tag{4.17}$$

Směrodatná odchylka $\sqrt{D\bar{\xi}}$ pak bude úměrná druhé odmocnině z počtu pozorování n.

Centrální limitní věta počtu pravděpodobnosti

Budou-li $\xi_1, \xi_2, \ldots, \xi_n$ nezávislé náhodné veličiny se stejným rozdělením, které má střední hodnotu μ a rozptyl σ^2, pak jejich součet má pro velká n přibližně Gaussovo rozdělení s parametry $N(n\,\mu, n\,\sigma^2)$.

4.2.5 Statistické testování hypotéz

Při analýze experimentálních dat se často setkáváme s úlohou rozhodnout, zda je možno změřené údaje popsat námi zvoleným teoretickým modelem nebo rozhodnout mezi několika navrhovanými modely, a případně z těchto dat určit i parametry vybraného rozdělení. Nejvýznamnější z testů, které se k tomuto účelu používají, je *test dobré shody*, často známý jako χ^2 *test*.

> Základem je spojitá náhodná veličina χ^2_f, která je zavedena následujícím způsobem: Vezmeme nezávislé náhodné veličiny $\zeta_1, \ldots, \zeta_f$, každou s Gaussovým rozdělením $N(0,1)$. Veličinu χ^2_f zkonstruujeme podle vztahu
>
> $$\chi^2_f = \sum_{i=1}^{f} \zeta_i^2\,.$$

Číslo f značící počet sčítanců se nazývá stupněm volnosti rozdělení. Momenty veličiny χ^2_f jsou se stupněm volnosti těsně svázány

$$\begin{aligned} E\chi^2_f &= f \\ D\chi^2_f &= 2f. \end{aligned}$$

Při praktické realizaci testu dobré shody je třeba mít tabulky veličiny χ^2_f – konkrétně se používají tzv. kritické hodnoty rozdělení $\chi^2_f(q)$, což nejsou náhodné veličiny, ale čísla. Jsou definovány vztahem

$$\int_{\chi^2_f(q)}^{\infty} p(\chi^2_f)\ \mathrm{d}\chi^2_f = \frac{q}{100}\,,$$

kde $q/100$ značí pravděpodobnost – bude vysvětleno dále.

Uveďme si jako příklad několik členů z tabulky kritických hodnot rozdělení $\chi^2_f(q)$:

q	**3**	**5**	**10**	**15**	**20**	**30**
99,9	0,024	0,210	1,48	3,48	5,92	11,6
99,0	0,115	0,554	2,56	5,23	8,26	15,0
90,0	0,584	1,61	4,87	8,55	12,4	20,6
...	...	...	...	...	...	...
10,0	6,25	9,24	16,0	22,3	28,4	40,3
1,0	11,3	15,1	23,2	30,6	37,6	50,9
0,1	16,3	20,5	29,6	37,7	45,3	59,7

Tabulka 4.1: Kritické hodnoty rozdělení χ^2_f v závislosti na počtu stupňů volnosti $f = 3, 5, 10, 15, 20, 30$ a pravděpodobnosti (riziku) q vyjádřené v procentech).

Při vlastním χ^2 testu budeme postupovat takto: Budeme-li mít n experimentálních údajů, rozdělíme je do k skupin, tzv. tříd. V každé třídě bude n_i údajů (empirická četnost). Navrhneme určitý teoretický model pro rozdělení změřených dat jenž budeme chtít testovat a na jeho základě určíme pravděpodobnosti $p_1, \ldots, p_k$, $\sum p_i = 1$, toho, že údaj padne do třídy i. Pak zkonstruujeme náhodnou veličinu

$$\chi^2 = \sum_{i=1}^{k} \frac{(n_i - np_i)^2}{np_i}\,. \tag{4.18}$$

Lze ukázat, že tato veličina má při $n \to \infty$ asymptotické rozdělení χ^2_{k-1}. V praxi se veličina χ^2_{k-1} používá jako aproximace rozdělení

χ^2, budou-li všechny hodnoty np_i (tzv. teoretické četnosti) větší nebo rovny 5. Vysoká přesnost závěrů je pak zaručena pro $np_i > 20$. Splnění těchto podmínek zajistíme vhodnou volbou tříd tak, aby teoretická pravděpodobnost p_i nebyla v žádné třídě příliš malá.

Pokud se nyní vrátíme k náhodné veličině χ^2_f, tak zjistíme, že kritické hodnoty $\chi^2_f(q)$ jsou takové hodnoty, které náhodná veličina s rozdělením χ^2 a f stupni volnosti překročí s pravděpodobností $q/100$

$$P\{\chi^2 \geq \chi^2_f(q)\} = \frac{q}{100}\,.$$

Ověřování statistické hypotézy se známými parametry

V případě, že nepotřebujeme určovat parametry našeho předpokládaného rozdělení a stačí nám pouze určit, zda naše hypotéza neodporuje experimentálním faktům, spočteme hodnotu χ^2 podle vztahu (4.18). Budeme-li mít data rozdělena do k tříd, zvolíme stupeň volnosti f podle vztahu

$$f = k - 1.$$

Při vlastním rozhodování ale musíme ještě zvolit *riziko* omylu, které jsme ještě ochotni připustit. Nejčastěji bývá voleno riziko v rozmezí 0,05 (tj. $q = 5$ procent) až 0,01 ($q = 0,1$ procenta).

Čím bude větší odchylka naší hypotézy od experimentálních údajů (tj. odchylka hodnot np_i od n_i), tím bude spočítaná hodnota χ^2 větší. Pomocí hodnot f (sloupec tabulky kritických hodnot) a q (řádek tabulky) určíme příslušnou kritickou hodnotu $\chi^2_f(q)$. Hypotézu přijmeme tehdy, když spočítaná hodnota χ^2 bude menší než nalezená kritická hodnota

$$\chi^2 < \chi^2_f(q) \tag{4.19}$$

a naopak, hypotézu jsme oprávněni vyloučit (s rizikem omylu $q/100$) jen tehdy, když námi určená hodnota χ^2 překročí kritickou hodnotu. Tedy, čím budeme úzkostlivější a budeme volit menší riziko, tím – jak je vidět z tabulky 4.1 – dostaneme větší kritické hodnoty a naší hypotézu budeme moci vyloučit jen při skutečně velkém nesouladu údajů.

Ověřování statistické hypotézy s neznámými parametry

Pokud budeme mít hypotézu s jedním nebo více volitelnými parametry – např. budeme předpokládat, že studovaný proces je poissonovský a nebudeme znát hodnotu parametru λ – musíme nejprve tyto parametry určit. Nejčastěji se k tomu využijí právě naměřené údaje. Kdybychom však potom použili test dobré shody ve výše popsané podobě,

došli bychom k nesprávným závěrům, neboť fitováním parametrů jsme celý proces ovlivnili.

Abychom toto ovlivnění vykompenzovali, musíme stupeň volnosti f zvolit podle modifikovaného vztahu

$$f = k - 1 - m\,,$$

kde m je počet parametrů našeho teoretického rozdělení, které jsme určovali na základě používaných experimentálních údajů. Dále lze použít χ^2 test již obvyklým způsobem.

V experimentální fyzice budeme nejčastěji pracovat s hypotézou o tom, že analyzovaný proces je gaussovský. I zde lze s úspěchem používat χ^2 test. Pokud ale nebudeme znát parametry Gaussova rozdělení μ a σ předem a budeme je chtít fitovat, bude celý proces dosti pracný. Proto byly navrženy silnější testy než χ^2 test, především Studentův test, které by bylo vhodnější použít – např. [8], atd.

Úprava experimentálních dat

Prozatím jsme χ^2 test používali na testování hypotéz, které měly za cíl vysvětlit naše měření, přičemž o správnosti těchto experimentálních dat jsme nepochybovali. Ve skutečnosti jsou ale experimentální údaje nepřesné:
- jsou zatížené šumem
- mohou být přítomné tzv. *výpadky.*

Že je v datech přítomný šum a to gaussovské povahy, s tím se v χ^2 testu počítá a naopak na tomto předpokladu je test založen. Naproti tomu výpadky nelze předem předvídat a do testu zahrnout, neboť vznikají nepředvídatelným způsobem – např. v průběhu našeho měření byl v sousední laboratoři zapnut přístroj, který se stal krátkodobým zdrojem silného rušení a tak byl jeden údaj nebo malá skupina údajů z našeho měření zkreslen. Ale tento nesprávný údaj nebo údaje ovlivní vyhodnocení celého měření, proto jediný správný způsob práce s výpadky je jejich vyloučení ze sady experimentálních dat.

Problém je ale v tom, jak výpadek odlišit od velké fluktuace experimentálních dat způsobené přirozeným šumem. Pokud bychom automaticky vyloučili všechny příliš odlišné údaje, existuje riziko, že současně odstraníme nějaký podstatný rys našeho studovaného procesu. K tomu, abychom dokázali (opět s předem zvoleným rizikem) rozlišit výpadek a šum, existuje více metod (např. pomocí integrálních transformací – viz druhý díl tohoto studijního textu) a jednou z nich je také χ^2 test.

Předpokládejme, že jako výsledek nějakého měření dostaneme posloupnost hodnot $\xi_1, \ldots, \xi_n$ a že tyto veličiny mají normální rozdělení.

Nechť cílem našeho měření je určit střední hodnotu této gaussovské veličiny $E\xi = \mu$. Při použití χ^2 testu na odstranění případných výpadků z experimentálních údajů $\xi_1, \ldots, \xi_n$ musíme rozlišit tři případy, jež mohou nastat:

1. Známe oba parametry Gaussova rozdělení
   Pokud předem známe oba parametry Gaussova rozdělení popisujícího naše naměřená data μ a σ, tj. provádíme pouze kontrolu správnosti těchto dat s cílem vyloučení výpadků, budeme postupovat takto:
   - Vezmeme nejvíce odlišný změřený údaj $\xi^* = max(\xi_1, \ldots, \xi_n)$, resp. $\xi^* = min(\xi_1, \ldots, \xi_n)$.
   - Spočítáme pravděpodobnost
     $$P\{|\xi^* - \mu| \geq k \cdot \sigma\} = 1 - \Phi^n(k),$$
     kde $\Phi(k)$ je tabelovaný integrál pravděpodobnosti.
   - Pokud tato pravděpodobnost bude menší než předem zvolené riziko $q/100$, údaj můžeme ze souboru dat vypustit.
   - Celý postup podle potřeby opakujeme.
2. Známe pouze rozptyl měření
   Pokud předem známe pouze parametr σ, budeme postupovat takto:
   - Spočítáme aritmetický průměr všech naměřených dat $\bar{\xi}$.
   - Najdeme nejvíce odlišný údaj ξ^*.
   - Vyšetříme opět podmínku pro vypuštění tohoto údaje, jež bude ale nyní podstatně složitější
     $$-\frac{n^2}{2} \cdot \left[1 - \Phi\left(k \cdot \sqrt{\frac{n}{n-1}}\right)\right]^2 + n \cdot \left[1 - \Phi\left(k \cdot \sqrt{\frac{n}{n-1}}\right)\right]$$
     $$\leq\ P\{\frac{1}{\sigma}|\xi^* - \bar{\xi}| \geq k\}\ \leq\ n \cdot \left[1 - \Phi\left(k \cdot \sqrt{\frac{n}{n-1}}\right)\right].$$
3. Neznáme předem žádný parametr Gaussova rozdělení
   V tomto případě nelze χ^2 test použít a musíme pracovat se silnějším Studentovým rozdělením [8].

Ve výše uvedených testech jsme používali integrál pravděpodobnosti $\Phi(x)$. Tato funkce vznikla integrací normovaného Gaussova rozdělení $N(0, 1)$, tj. je jeho distribuční funkcí

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{x} \exp\left(-\frac{t^2}{2}\right) \mathrm{d}t \tag{4.20}$$

a patří mezi standardní funkce matematické statistiky. Její tabelované hodnoty lze nalézt v jakýchkoliv statistických tabulkách, případně si ji lze snadno naprogramovat.

4.2.6 Entropie a informace

Pro popis stavu systému a jeho změn se v termodynamice používá veličina zvaná *entropie*. Tuto veličinu zavedl Clausius již v roce 1850 [9] a s její pomocí lze v termodynamice charakterizovat vývoj systémů, skupenské přeměny, atd. Pro nevratné děje probíhající v přírodě bylo nalezeno, že každá samovolná přeměna je doprovázena výsledným vzrůstem entropie.

V současné době je pojem entropie široce využíván v teorii informace.

Míru informace lze kvantitativně popsat: Mějme proces, který může nabývat N různých hodnot, všechny se stejnou pravděpodobností. Pokud se dozvíme, která realizace skutečně nastala, získali jsme informaci I, jejíž míru můžeme definovat vztahem

$$I = K \ln N \,.$$

Podle tohoto vztahu získáme tím více informace, čím více bylo původních hodnot N, jejichž seznam jsme v průběhu získávání konkrétní informace zúžili na jednu. Funkce logaritmus se používá proto, že informace je aditivní, zatímco počet jevů narůstá geometrickou řadou – máme-li dvě nezávislé soustavy s počtem možných hodnot N_1 a N_2, celkový počet hodnot v kombinaci obou soustav bude $N_1 \cdot N_2$, zatímco výsledná informace o kombinované soustavě I bude rovna $I_1 + I_2$.

Konstanta K určuje jednotku informace. V binární soustavě, kdy jednotkou informace je jeden bit, volíme K tak, abychom mohli informaci definovat vztahem

$$I = \log_2 N \,.$$

Chceme-li najít souvislost mezi teorií informace a termodynamikou, musíme informaci měřit v termodynamických jednotkách. V tomto případě položíme konstantu K rovnou Boltzmannově konstantě k_B. Budeme-li sledovat vývoj soustavy, potom pokud jsme měli na počátku informaci I_0 odpovídající N_0 možnostem a na konci informaci $I_1 > 0$ odpovídající N_1 možnostem, platí

$$I_1 = k_B \ln \frac{N_0}{N_1} \,.$$

Při přechodu na termodynamický popis téhož procesu [9] pomocí entropie S dostaneme výraz $\Delta S = S_1 - S_0 = k_B \ln(N_1/N_0)$, neboli

$$-I_1 = S_1 - S_0 = \Delta S \,.$$

Proto podle terminologie teorie informace entropie představuje negativní informaci, tj. při vzrůstu informace o soustavě dochází k poklesu entropie soustavy.

4.3 Základní techniky metody Monte Carlo

V sekci 4.1 jsme si uvedli základní schéma řešení problémů metodou Monte Carlo. Toto schéma se skládalo ze čtyř kroků. První z nich je nejobtížnější – vytvoření vhodného modelu, ostatní tři kroky již představují rutinní postupy, i když i tam lze v konkrétních případech narazit na problémy.

Čtvrtý krok – statistické vyhodnocení výsledků – bychom měli být již schopni vyřešit na základě poznatků z počtu pravděpodobnosti a matematické statistiky, které jsme si uvedli v předchozí sekci. Modely jsou většinou formulovány tak, že pokud hledáme jednočíselný výsledek, nalezneme jej jako první nebo druhý moment námi zvolené náhodné veličiny ξ, $E\xi$ nebo $D\xi$, a tyto momenty umíme podle zákona velkých čísel přibližně odhadnout. Pokud bude řešením našeho problému nějaké rozdělení (např. úhlové nebo energetické rozdělení částic), snažíme se obvykle model vytvořit tak, aby hledané rozdělení představovalo rozdělení pravděpodobností naší náhodné veličiny a to již z posloupnosti $\xi_1, \xi_2, \ldots, \xi_N$ přibližně zrekonstruujeme.

Nyní si probereme, jak budeme realizovat druhý a třetí krok základního schématu.

4.3.1 Generování náhodných čísel

Výpočetní schéma metody Monte Carlo předpokládá modelování náhodného procesu pomocí operací s náhodnými čísly. Tato čísla je nutno v deterministickém prostředí počítače získat speciálním způsobem. Během krátké historie metody Monte Carlo byly navrženy a použity tyto způsoby:

- Fyzikální generátory
- Tabulky náhodných čísel
- Vypočítaná náhodná čísla

V počátcích byla známa jediná technika – použití nějakého fyzického generátoru, tj. samostatného zařízení produkujícího náhodné veličiny, a přenesení takto vytvořených čísel do počítače. Takovým zařízením byla nejprve ruleta (odtud dostala metoda Monte Carlo své jméno), později byly používány systémy přímo elektricky propojené s počítačem, které využívaly fyzikální procesy náhodné již svou podstatou. Obvykle to byl buď radioaktivní rozpad nebo tzv. výstřelový šum v elektronkách. Tyto generátory byly používány až do začátku šedesátých let. Produkovaly skutečně náhodná čísla a jediným důvodem, proč byly opuštěny, byla jejich relativní pomalost – výkonnost počítačů se zvyšovala rychleji než to dokázala sledovat měřící technika, na níž byla tato první generace generátorů založena, viz tabulka 2.1.

Další metodou, která se používala několik let, byly tabulky náhodných čísel, tj. fyzikální generátory vyprodukovaly náhodná čísla „do zásoby“, tato čísla se uložila na nějaké vnější médium a při modelování se pak využila. V šedesátých letech dostupnými médii byly buď tabulky vytištěné na papíře a nebo děrná či magnetická páska. Rychlost výpočetní techniky se však zvyšovala tak rychle, že toto řešení brzy přestalo vyhovovat.

Řešení, které bylo zvoleno jako náhrada za fyzikální generátory byla tzv. „vypočítaná náhodná čísla“. Toto řešení se používá až do současné doby. Již sám pojem vypočítaná náhodná čísla v sobě obsahuje protimluv – náhodnost nelze žádným deterministickým postupem vypočítat:

„Každý, kdo se zabývá aritmetickými metodami vytváření náhodných čísel, se nepochybně dopouští hříchu.“

John von Neumann (1951)

Ve skutečnosti se jedná o algoritmy, které vedou na posloupnosti čísel více nebo méně se podobající číslům náhodným. Proto tato čísla správně nazýváme *pseudonáhodná*. Od skutečně náhodných čísel se odlišují ve dvou rysech:

– Posloupnost nagenerovaných pseudonáhodných čísel je vždy konečná, po uplynutí tzv. periody p se začne celá posloupnost opakovat.

– V rámci jedné periody se čísla vyskytují relativně náhodně, ale tato náhodnost není dokonalá.

Cílem teorie generátorů pseudonáhodných čísel je vybrat algoritmus jejich generování tak, aby oba rysy co nejméně rušily – tj. vytvořit posloupnosti s periodou výrazně delší (aspoň desetkrát) než je počet náhodných čísel používaných v jednom počítačovém experimentu a v rámci jedné periody mít čísla rozdělena co nejnáhodněji. Vytvoření kvalitního generátoru pseudonáhodných čísel není totiž vůbec jednoduchá záležitost. Generátor je nejprve vybrán na základě nějakých teoretických předpokladů a pak dlouhodobě velice důkladně testován – např. zda jsou v něm náhodně generovány jednotlivé číslice, pak dvojice, trojice, atd. a zda frekvence výskytu určitých kombinací číslic (např. 222 nebo 12345, ...) odpovídá teorii. Při těchto statistických testech kvality generátorů se používá χ^2 test (viz sekce 4.2). Touto problematikou se vážně zabývají profesionální matematici a v prvním přiblížení lze proto předpokládat, že generátory zabudované do překladačů hlavních programovacích jazyků dodávaných špičkovými softwarovými firmami oba tyto požadavky v rozumné míře splňují (i když již v minulosti došlo k výrazným zklamáním).

Postupně byla vytvořena celá řada algoritmů, jejichž použití dává posloupnost pseudonáhodných čísel s dobrými statistickými vlastnostmi. Některé algoritmy vedou na rekurentní vzorce prvního řádu

$$\xi_{i+1} = a\xi_i + b \quad (mod\ M),$$

častěji však jsou používány rekurentní vzorce vyšších řádů, které nové pseudonáhodné číslo ξ_{i+1} určují na základě operací s $k+1$ předchozími čísly ξ_i, ξ_{i-1} až ξ_{i-k}. Perioda generátoru obecně závisí na jeho řádu k podle vztahu $p \sim M^k$. Převážná většina generátorů pseudonáhodných čísel vytváří jednotlivé hodnoty náhodné veličiny γ rovnoměrně rozdělené v intervalu (0,1). V tom, zda hodnoty 0 a 1 patří do definičního intervalu, se generátory liší a před použitím konkrétního generátoru ve vlastním programu to je třeba zjistit, neboť chyby způsobené např. nagenerováním náhodného čísla s nulovou hodnotou v programu, který s touto hodnotou provádí operaci dělení, se při ladění programu jen velmi obtížně hledají.

Ve skutečnosti existují dvě třídy vypočítaných náhodných čísel:

- *čísla pseudonáhodná*, a
- *čísla kvazináhodná*.

Obě skupiny generátorů se snaží vytvořit náhodná čísla γ rovnoměrně rozdělená na intervalu (0,1), ale výrazně se liší ve stupni náhodnosti vytvořené posloupnosti.

Mnohem běžnější jsou čísla pseudonáhodná – pokud se ve fyzikální laboratoři mluví o náhodných číslech, téměř vždy se tím rozumějí čísla pseudonáhodná. *Generátory pseudonáhodných čísel* jsou založeny na jednom ze tří principů:

- von Neumannovy generátory,
- lineární kongruenční generátory, nebo
- generátory s posuvnými registry.

Von Neumannovy generátory se používaly jen v samých počátcích (od roku 1946) a nyní mají pouze historickou cenu, neboť jejich periody jsou příliš krátké a náhodnost nedostatečná. Tyto generátory byly založeny na manipulaci s jednotlivými číslicemi dekadických čísel.

Nejpoužívanější v naší době jsou lineární kongruenční generátory. Jsou založené na vztahu pro nalezení nového pseudonáhodného čísla ξ_{i+1} na základě již známých čísel ξ_i až ξ_{i-k}

$$\xi_{i+1} = a_0 \cdot \xi_i + a_1 \cdot \xi_{i-1} + \ldots + a_k \cdot \xi_{i-k} + b \quad (mod\ M), \qquad (4.21)$$

kde $k \geq 0$, a_k, b a M jsou konstanty, jejichž vhodnou volbou určujeme konkrétní generátor. Díky operaci „modulo“ leží vytvořená pseudonáhodná čísla v intervalu 0 až $M-1$. Na obvyklý interval 0 až 1 je převedeme vydělením hodnotou M – takto vzniklé číslo γ leží v intervalu $\langle 0, 1)$. Abychom takový generátor nastartovali, musíme použít tzv. násadu, tj. hodnoty čísel ξ_i, ξ_{i-1} až ξ_{i-k}, kde všechna čísla násady musí být menší než hodnota M. Pokud tuto násadu nebudeme měnit, dostaneme vždy stejnou posloupnost pseudonáhodných čísel. Tato okolnost, která by byla škodlivá při skutečném výpočtu, se využívá při ladění programu. Další okolnost, kterou musíme při aplikaci konkrétního generátoru vzít v úvahu je, že obvykle počátek nagenerované posloupnosti pseudonáhodných čísel není dosti náhodný – příliš závisí na použité násadě. Proto se provádí tzv. zahřívání generátoru, kdy se prvních několik desítek nebo set nagenerovaných čísel nevyužije.

Konkretizací konstant v definičním vztahu pro lineární kongruenční generátor můžeme vytvořit tři zjednodušené typy generátorů – aditivní, multiplikativní a smíšený.

Multiplikativní generátory (známé od roku 1951, autor Lehmer) jsou založené na rekurentním vztahu prvního řádu s nulovou hodnotou konstanty b

$$\xi_{i+1} = a_0 \cdot \xi_i \quad (mod\ M). \tag{4.22}$$

Tyto generátory jsou dostatečně rychlé, ale jejich kvalitu je nutno důkladně zvažovat. Jako příklad si můžeme uvést

$$\xi_{i+1} = 7^5 \cdot \xi_i \quad (mod\ 2^{31} - 1)$$

$$\xi_{i+1} = 630\,360\,016 \cdot \xi_i \quad (mod\ 2^{31} - 1).$$

Oba tyto generátory používala v 70. a 80. letech firma IBM u svých počítačů řad IBM 360 a IBM 370. Další generátor

$$\xi_{i+1} = 13^{13} \cdot \xi_i \quad (mod\ 2^{59})$$

používala firma NAG ve své knihovně podprogramů jazyka FORTRAN. Pokud použijeme nevhodnou volbu konstant b a M, kvalita generátoru velmi prudce klesá. Jako příklad nevhodného generátoru tohoto typu si uveďme generátor použitý v překladači jazyka BASIC osmibitového mikropočítače Sinclair ZX Spectrum

$$\xi_{i+1} = 75 \cdot \xi_i \quad (mod\ 65\,537).$$

Perioda tohoto generátoru je navíc velmi krátká – má hodnotu pouze 65 538.

Aditivní generátory jsou založené na vztahu

$$\xi_{i+1} = \xi_{i-k1} + \xi_{i-k2} \quad (mod\ M). \tag{4.23}$$

Příklad velmi jednoduchého generátoru tohoto typu je

$$\xi_{i+1} = \xi_{i-1} + \xi_{i-2} \quad (mod\ 3\,137).$$

Generátor je dosti kvalitní, velmi rychlý (operace modulo se provede prostým testem, zda vzniklé číslo je menší než M a pokud ne, tak se M odečte). Slabinou je pro naši dobu již velmi krátká perioda, přibližně jen $9,8 \cdot 10^9$ (tj. M^2). Generátor je proto vhodný zejména pro ladění. Vylepšená varianta tohoto generátoru se však používá profesionálně

$$\xi_i = \xi_{i-5} + \xi_{i-17} \quad (mod\ M),$$

kde perioda je závislá na hodnotě modulu – pro $M = 2^8$ je $p = 1,6 \cdot 10^7$, pro $M = 2^{16}$ je $p = 4,3 \cdot 10^9$ a pro $M = 2^{32}$ je $p = 2,8 \cdot 10^{14}$. Další výhodou, kromě dlouhé periody, je tvar konstanty M – ve strojovém kódu se operace modulo u 8-, 16- nebo 32-bitových proměnných realizuje automaticky prostým odříznutím přebývající částí čísla.

Smíšené generátory jsou založené na lineárním rekurentním vztahu

$$\xi_{i+1} = a \cdot \xi_i + b \quad (mod\ M). \tag{4.24}$$

Příkladem jsou generátory

$$\xi_{i+1} = 69\,069 \cdot \xi_i + 1 \quad (mod\ 2^{32})$$

nebo

$$\xi_{i+1} = 5^{17} \cdot \xi_i + 1 \quad (mod\ 2^{48}).$$

Generátory s posuvnými registry se v uživatelských programech používají jen výjimečně, neboť jsou svou podstatou předurčeny pro přímou hardwarovou realizaci – existují jednoúčelové počítače obsahující integrovaný obvod přímo generující pseudonáhodná čísla a takový obvod je konstruován podle algoritmu pro generátor s posuvným registrem

$$b_{i+1} = c_0 \cdot b_i + c_1 \cdot b_{i-1} + \ldots + c_k \cdot b_{i-k} \quad (mod\ 2). \tag{4.25}$$

Tento vztah je velmi podobný obecnému vztahu pro lineární kongruenční generátor, základní rozdíl je ale ve významu konstant: $k \geq 1$, c_k nabývají pouze hodnot 0 nebo 1 (nejméně dvě hodnoty musí být nenulové) a generátor pracuje místo s celými čísly pouze s bity b. Navíc, při praktickém použití se pro zvýšení kvality generátoru volí konstanta k

(a tím i velikost násady) mnohem větší než u lineárních kongruenčních generátorů. Při praktické aplikaci se často pracuje s lineárními generátory, kde pouze dvě hodnoty c jsou nenulové, např.

$$b_{i+1} = b_{i-470} + b_{i-9688} \quad (mod\ 2),$$

kde vstupní posloupnost bitů musí mít délku 9 689 – tento vztah lze s pomocí logické operace XOR též zapsat jako

$$b_{i+1} = b_{i-470}\ XOR\ b_{i-9688}.$$

Perioda takového generátoru je skutečně mimořádná, může dosahovat hodnoty až $2^{9\,689} - 1$. Je samozřejmě možné i tento generátor naprogramovat, pro dostatečnou efektivitu výsledného kódu je ale třeba mít programovací jazyk umožňující přímou manipulaci s bity. V případě, že by byla potřeba skutečně takový generátor připravit na softwarové úrovni, existují i algoritmy s délkou bitů menší než 1 000 (tím samozřejmě klesne i perioda, přesto však bude více než dostatečná).

I u generátorů s posuvnými registry existuje několik typů, které však zde již nebudeme probírat. Zájemce lze odkázat na literaturu [10], případně na Internet.

Existují i snahy kvalitu generátorů pseudonáhodných čísel zvýšit, a to jak jejich periodu tak i náhodnost. Za tímto účelem lze kombinovat více generátorů, pokud možno založených na různých typech algoritmů. Jednou z možných cest je jedním generátorem naplnit větší množinu pseudonáhodných čísel a druhým generátorem z této množiny „losovat“ konkrétní číslo. První generátor pak nahradí již použité číslo novým a postup se opakuje. Tento postup sice zvýší náhodnost generované posloupnosti čísel, současně však zpomaluje výpočet a prakticky znemožňuje teoretickou analýzu generovaných výsledků. Proto má tento postup své příznivce i odpůrce.

Druhou skupinou vypočítaných náhodných čísel jsou *čísla kvazináhodná*. Posloupnost těchto čísel se vůbec nepodobá náhodným číslům, přesto však nacházejí v počítačové fyzice své uplatnění.

Základní myšlenka generátorů kvazináhodných čísel je založena na požadavku co nejrovnoměrněji pokrýt interval (0,1). I skutečná náhodná čísla pokrývají úsečku (0,1) relativně rovnoměrně, ale s náhodnými fluktuacemi, které zde přítomné nebudou.

Generátory kvazináhodných čísel mohou být založeny na libovolném prvočísle. Ukážeme si proto několik prvních hodnot posloupnosti kvazináhodných čísel generovaných na základě prvočísla 2:

$$1/2,\ 1/4,\ 3/4,\ 1/8,\ 5/8,\ 3/8,\ 7/8,\ 1/16,\ \ldots\ .$$

> Použití kvazináhodných čísel v počítačové fyzice je velmi omezené. Nelze je použít tam, kde požadujeme náhodnost a nezávislost dvou po sobě jdoucích náhodných čísel (např. souřadnice náhodného bodu), neboť kvazináhodná čísla jsou v tomto ohledu přísně determinovaná. Pokud ale požadujeme od generátoru náhodná čísla, která budou zpracovávána od sebe odděleně, projeví se výhoda neexistence fluktuací v posloupnosti kvazináhodných čísel. Uvedli jsme si empirický vztah (4.1) pro chybu metody Monte Carlo a jeho ověření na základě matematické statistiky, podle nichž metoda konverguje jako $1/\sqrt{N}$. Tento vztah je platný při použití skutečně náhodných i pseudonáhodných čísel, pouze při použití kvazináhodných čísel se konvergence metody Monte Carlo zrychluje na téměř $1/N$.

I když v naší práci budeme prakticky vždy pracovat s pseudonáhodnými čísly, vývoj pokračuje dále. V 90. letech byla vytvořena nová generace fyzikálních generátorů skutečně náhodných čísel s využitím nových principů experimentální fyziky. Tyto generátory vyhovují svou rychlostí i současným superpočítačům, jejich cena však tomu odpovídá a nejsou proto použitelné v běžné laboratoři (uvažuje se však o vytvoření zjednodušené varianty výkonem i cenou odpovídající obvyklým mikropočítačům nebo pracovním stanicím). Též myšlenka na vytváření tabulek náhodných čísel se vrací v moderní verzi s využitím současných rychlých velkokapacitních médií – CD-ROM a DVD-ROM.

4.3.2 Transformace náhodných veličin

Dalším krokem výpočetního schématu metody Monte Carlo je transformace náhodné veličiny γ s rovnoměrným rozdělením na intervalu (0,1), kterou nám poskytuje generátor náhodných čísel, na libovolnou náhodnou veličinu ξ, již jsme zvolili v rámci vytváření našeho modelu.

Existuje velké množství algoritmů, které k tomuto účelu byly navrženy. Můžeme si je rozdělit na dvě skupiny:

– algoritmy obecné, jimiž lze vyřešit větší množství problémů, ale případně s nízkou efektivitou, a

– algoritmy specializované pro konkrétní náhodné veličiny ξ (a právě těchto druhých algoritmů je veliký počet).

Předpokládejme, že máme k dispozici spojitou náhodnou veličinu γ rovnoměrně rozdělenou na intervalu (0,1). Parametry jejího rozdělení jsou

$$E\gamma \;=\; \frac{1}{2}$$

$$D\gamma = \frac{1}{12}.$$

Při konkrétní realizaci na počítači získáme ve skutečnosti m-místná diskrétní čísla γ', která budou mít poněkud změněné momenty

$$\begin{aligned} E\gamma' &= \frac{1}{2} \cdot (1 - 2^{-m}) \\ D\gamma' &= \frac{1}{12} \cdot (1 - 2^{-2m}). \end{aligned}$$

Pokud budeme pracovat s dostatečně-místnou aritmetikou, můžeme většinou rozdíly mezi veličinami γ a γ' zanedbat a výstup z generátoru náhodných čísel podle potřeby považovat buď za diskrétní a nebo za spojitou náhodnou veličinu.

Z obecných algoritmů pro transformaci náhodných veličin (v terminologii metody Monte Carlo se běžně používá pojem „rozehrání“) si uvedeme následující čtyři:

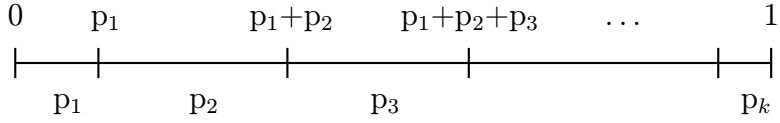
1. Rozehrání diskrétní náhodné veličiny
   V tomto případě budeme mít veličinu ξ zadánu tabulkou

$$\xi = \begin{pmatrix} x_1 & x_2 & \dots & x_k \\ p_1 & p_2 & \dots & p_k \end{pmatrix}$$

   V počítači si vytvoříme vektor o k složkách: $p_1, p_1 + p_2, p_1 + p_2 + p_3, \dots, p_1+p_2+\dots+p_{k-1}, 1$ (obr. 4.2). Pak nageneruje me jednu hodnotu γ a určíme, do kterého intervalu padne, tj. budeme postupně vyšetřovat podmínku

$$\gamma < \sum_{i=1}^{j} p_i .$$

   První interval j, pro který bude tato podmínka splněna, určí příslušnou hodnotu $\xi = x_j$.



Obrázek 4.2: Rozložení pravděpodobností p_i na intervalu $\langle 0, 1 \rangle$.

Tento algoritmus lze použít pro transformaci libovolné diskrétní náhodné veličiny, proto další metody již nejsou potřeba. Pouze v případě, že počet možných hodnot veličiny ξ, k, je příliš velký, takže by vzestupné hledání příslušného intervalu j trvalo příliš dlouho, lze použít rychlejší postup – např. metodu půlení intervalu. Princip metody transformace diskrétní náhodné veličiny se tím však nezmění.

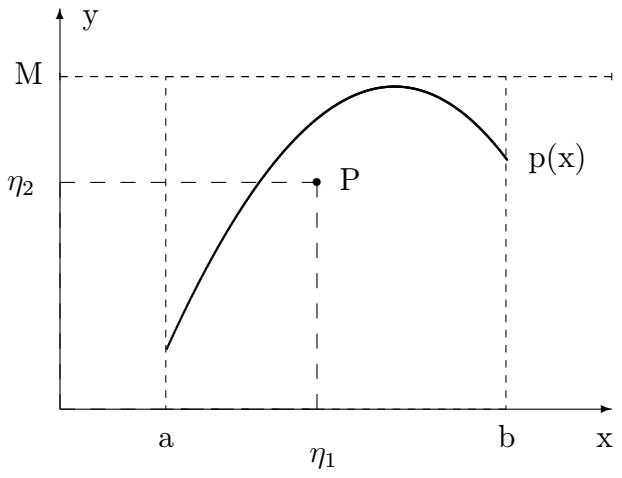
2. Rozehrání spojité náhodné veličiny - metoda inverzní funkce
V případě spojitých náhodných veličin máme algoritmů více. Nejjednodušším z nich je metoda inverzní funkce. Použijeme-li obvyklé značení pro hledanou spojitou náhodnou veličinu ξ:

$$\text{obor hodnot } \langle a, b\rangle, \text{ hustota pravděpodobnosti } p(x),$$

musíme pro nalezení hodnoty spojité náhodné veličiny ξ na základě veličiny γ vyřešit rovnici

$$\int_a^{\xi} p(x)\,\mathrm{d}x = \gamma\,. \tag{4.26}$$

Výsledkem bude transformační vztah typu $\xi = g(\gamma)$. Rovnice, již musíme řešit, je však implicitní pro veličinu ξ a analytické řešení této rovnice nemusí existovat. Pak je metoda inverzní funkce nepoužitelná a zkusíme další metody.



Obrázek 4.3: Metoda výběru pro generování náhodné veličiny ξ.

3. Rozehrání spojité náhodné veličiny - metoda výběru
Metoda se také nazývá metoda von Neumannova. Pro její použití stačí předpoklad, že hustota pravděpodobnosti $p(x)$ veličiny ξ je omezená na intervalu $\langle a, b\rangle$. Postup použití metody je následující (obr. 4.3):

- Zvolíme konstantu M tak, aby platilo $p(x) \leq M, x \in \langle a, b \rangle$.
- Nagenerujeme dvě hodnoty náhodné veličiny γ, γ_1 a γ_2, a vytvoříme čísla $\eta_1 = a + \gamma_1(b - a)$, $\eta_2 = M\gamma_2$.
- Bude-li bod P o souřadnicích (η_1, η_2) ležet pod křivkou $y = p(x)$, tj. bude-li $\eta_2 < p(\eta_1)$, zvolíme $\xi = \eta_1$. Nebude-li podmínka splněna, nagenerujeme novou dvojici γ_1 a γ_2 a postup opakujeme.

Účinnost metody závisí na hodnotě M, kterou je nutno zvolit co nejmenší, tj. co nejblíže hodnotě $M = \sup p(x)$ pro $a \leq x \leq b$.

4. Metoda superpozice
   Tato metoda není omezena pouze na spojité náhodné veličiny, i když ji budeme využívat téměř výhradně pro ně. Budeme hledat náhodnou veličinu ξ s distribuční funkcí

$$F(x) = \sum_{i=1}^{m} c_i \cdot F_i(x),$$

   kde $F_i(x)$ jsou též distribuční funkce, $c_1 + \ldots + c_m = 1$ a všechna $c_i > 0$. Zavedeme diskrétní náhodnou veličinu η s rozdělením

$$\eta = \begin{pmatrix} 1 & 2 & \ldots & m \\ c_1 & c_2 & \ldots & c_m \end{pmatrix}.$$

   Hledanou veličinu ξ s distribuční funkcí $F(x)$ najdeme tímto způsobem:

- nagenerujeme dvě nezávislé hodnoty γ_1 a γ_2 veličiny γ
- rozehrajeme číslem γ_1 hodnotu $\eta = k$
- z rovnice $F_k(\xi) = \gamma_2$ určíme veličinu ξ.

Pomocí kombinace výše uvedených čtyř algoritmů můžeme rozehrát jakoukoliv náhodnou veličinu, pro některé problémy je ale výrazně efektivnější používat specializované algoritmy. Uveďme si několik z nich, velký počet dalších lze nalézt v literatuře.

1. Modelování n-rozměrného náhodného bodu
   Jelikož jsou souřadnice náhodného bodu nezávislé, můžeme každou z nich modelovat zvláště pomocí kterékoliv z výše uvedených obecných metod. Pokud ale máme za úkol generovat body z oblasti složitého tvaru, je vhodné použít zobecněnou Neumannovu metodu výběru:
   Zvolíme jednoduchou oblast,která v sobě zadanou oblast obsahuje. Body generujeme v této nové větší oblasti a testujeme, padnou-li do naší původní oblasti. Tyto body zachováme a ostatní vyloučíme.

2. Rozehrání gaussovské náhodné veličiny ζ
   Požadujeme rozehrát náhodnou veličinu ζ s parametry

$$p(x) = \frac{1}{\sqrt{2\pi}} \cdot \exp(-\frac{x^2}{2}), \quad x \in (-\infty, \infty),$$

   tj. $N(0,1)$. Existuje více algoritmů, z nichž dva si uvedeme:

   (a) Zjednodušený postup je založen na centrální limitní větě počtu pravděpodobnosti. Hodnotu Gaussovské náhodné veličiny ζ dostaneme jako součet m hodnot rovnoměrně rozdělené veličiny γ

$$\zeta' \doteq \sum_{i=1}^{m} \gamma_i . \tag{4.27}$$

   Zkonstruovaná veličina ζ' bude mít parametry $\mu = \frac{m}{2}$, $\sigma = \frac{1}{2}\sqrt{\frac{m}{3}}$. Přechod na požadovanou normalizovanou veličinu ζ provedeme pomocí vztahu

$$\zeta = \frac{\zeta' - \mu}{\sigma} .$$

   Tento algoritmus je relativně rychlý, ale jeho přesnost je omezená – podle centrální limitní věty je zcela přesný jen pro $m \to \infty$. Minimálně použitelná veličina ζ se získá pro $m = 5$ a velmi vysoká přesnost je dosažena pro $m = 12$.

   (b) V některých aplikacích ale požadujeme zcela přesnou gaussovskou veličinu a pak můžeme použít např. vztah odvozený v [11]

$$\zeta = \sqrt{-2 \ln \gamma_1} \cdot \cos(2\,\pi\,\gamma_2), \tag{4.28}$$

   kde γ_1 a γ_2 jsou dvě hodnoty rovnoměrně rozdělené náhodné veličiny γ.

3. Modelování gamma rozdělení
   Požadujeme rozehrát náhodnou veličinu $\xi^{(n)}$ s parametry

$$p(x) = \frac{1}{(n-1)!} \cdot x^{(n-1)} \cdot \exp(-x), \quad x \in (0, \infty), \quad n \geq 1 \; \textit{celé}.$$

   K transformaci můžeme použít vzorec

$$\xi^{(n)} = -\ln(\gamma_1 \cdot \gamma_2 \cdot \ldots \cdot \gamma_n), \tag{4.29}$$

   kde pro generování gamma rozdělení n-tého řádu musíme použít n hodnot náhodné veličiny γ.

4. Rozehrávání náhodného bodu v kouli o poloměru R
   Budeme pracovat ve sférických souřadnicích (r, θ, φ). Transformační vztahy získané s použitím tří hodnot náhodné veličiny γ jsou

$$\begin{aligned} r &= R\sqrt[3]{\gamma_1} \\ \cos\theta &= 2\,\gamma_2 - 1 \\ \varphi &= 2\,\pi\,\gamma_3\,. \end{aligned} \tag{4.30}$$

   Pokud potřebujeme náhodný bod na povrchu koule, použijeme druhý a třetí vztah. Tyto dva vztahy jsou v počítačové fyzice velmi důležité, neboť s jejich pomocí rozehráváme náhodný směr – ten získáme pokud náhodným bodem na povrchu koule vedeme polopřímku ze středu koule.

4.3.3 Metoda Monte Carlo a matematická statistika

Nejběžnější formulací úloh řešených metodou Monte Carlo je hledání jedné neznámé hodnoty a. V tomto případě podle schématu metody vytvoříme náhodnou veličinu ξ tak, aby hledané číslo bylo prvním momentem této náhodné veličiny. Očekávanou hodnotu $E\xi$ a tím i hledané a odhadneme tak, že provedeme n nezávislých realizací náhodné veličiny ξ: $\xi_1, \ldots, \xi_n$. Podle zákona velkých čísel (4.16) platí, že aritmetický průměr těchto realizací $\bar{\xi}$ se pro dosti velká n blíží k a

$$a \doteq \frac{1}{n} \cdot \sum_{i=1}^{n} \xi_i\,.$$

Matematická statistika nám dává též informaci o chybě, které se dopouštíme, když očekávanou hodnotu $E\xi$ nahradíme aritmetickým průměrem $\bar{\xi}$. Budeme-li předpokládat, že veličina ξ má konečný rozptyl $D\xi$, dostaneme odhad

$$\lim_{n\to\infty} P\left\{-x < \frac{1}{\sqrt{n \cdot D\xi}} \cdot \sum_{i=1}^{n} (\xi_i - a) < x\right\} = 2\,\Phi(x) - 1\,,$$

kde $\Phi(x)$ je integrál pravděpodobnosti (4.20) – viz sekce 4.2.5.

x	0,675	1,212	1,645	1,96	2,58	3,0	3,29
$2\Phi(x) - 1$	0,50	0,80	0,90	0,95	0,99	0,997	0,999

Tabulka 4.2: Hodnoty upraveného integrálu pravděpodobnosti $2\,\Phi(x) - 1$.

Pro dosti velká n tedy přibližně platí

$$P\left\{|\bar{\xi} - a| < x \cdot \sqrt{\frac{D\xi}{n}}\right\} = 2\,\Phi(x) - 1\,.$$

Při použití tohoto vzorce pro odhad přesnosti simulace metodou Monte Carlo si nejprve stanovíme dovolené riziko $q/100$ a pak z tabulky integrálu pravděpodobnosti (Tab. 4.2) najdeme hodnotu x vyhovující vztahu

$$2\,\Phi(x) - 1 = 1 - q/100\,.$$

Jelikož náhodná veličina $\bar{\xi}$ je přibližně normální se směrodatnou odchylkou $\sigma = \sqrt{D\xi/n}$, na základě „pravidla tří sigma“ lze zvolit $1 - q/100 = 0,997$, čemuž odpovídá $x = 3,0$. Tato hodnota určuje horní odhad chyby. V reálných případech se obvykle volí mírnější podmínka – tzv. pravděpodobná chyba $1 - q/100 = 0,5$, čemuž odpovídá $x = 0,67$ a výsledný vztah pro chybu metody Monte Carlo bude mít tvar

$$\vartheta = 0,67 \cdot \sqrt{\frac{D\xi}{n}}\,. \tag{4.31}$$

Na základě tohoto vzorce můžeme porozumět empiricky nalezenému vztahu (4.1). Důležitý je poznatek, že chyba výpočtu metodou Monte Carlo závisí nejen na počtu pokusů, ale i charakteristice studované náhodné veličiny $D\xi$. To nám dává možnost pokusit se zlepšit přesnost výpočtu ne „hrubou silou“, tj. zvýšením počtu pokusů n, ale volbou modelu s menší disperzí $D\xi$.

Chybu simulace metodou Monte Carlo můžeme kontrolovat již v průběhu simulace a případně po dosažení požadované přesnosti řešení výpočet ukončit. K tomu využijeme vztah (4.7) pro rozptyl $D\xi = E(\xi^2) - (E\xi)^2$ spolu se zákonem velkých čísel (4.16). Znamená to v průběhu simulace v jedné proměnné postupně sčítat hodnoty ξ_i, v druhé proměnné ξ_i^2 a z nich průběžně konstruovat výraz

$$D\xi \doteq \frac{1}{n} \cdot \sum_{i=1}^{n} \xi_i^2 - \left[\frac{1}{n} \cdot \sum_{i=1}^{n} \xi_i\right]^2\,. \tag{4.32}$$

Přesnost tohoto odhadu, z nějž s využitím vztahu (4.31) můžeme přibližně určit chybu simulace, s rostoucí hodnotou n postupně narůstá.

Existuje i prostší metoda hrubého odhadu chyby simulace – k tomu stačí konstruovaný aritmetický průměr rozdělit na několik částí a tyto dílčí průměry navzájem porovnat.

4.4 Řešení numerických problémů pomocí metody Monte Carlo

Ačkoliv metoda Monte Carlo byla původně navržena pro řešení problémů z oblasti jaderné fyziky, ukázala se být velmi vhodnou pro řešení úloh ze všech vědních disciplin. Její použití se zdá být přirozené tam, kde sám studovaný problém má statistický charakter. Zajímavé však je, že se metoda Monte Carlo stejně dobře uplatňuje i v oblastech, kde se pro řešení úloh výhradně používají determinované algoritmy (příkladem může být i Buffonova jehla). Proto také se metoda Monte Carlo velmi dobře uplatňuje i v matematice, kde často nabízí velmi zajímavou alternativu ke klasickým numerickým algoritmům.

Algoritmy metody Monte Carlo byly rozpracovány pro výpočet jednoduchých i násobných integrálů, pro řešení soustav lineárních algebraických rovnic, pro operace s maticemi, pro řešení diferenciálních i integrálních rovnic, atd. S trochou nadsázky lze prohlásit, že ke každému algoritmu z klasické numerické matematiky lze najít ekvivalentní postup v rámci metody Monte Carlo. Otázkou pouze je, že zdaleka ne vždy jsou tyto alternativní postupy výhodnější.

V této sekci si proto uvedeme některé důležité příklady, kde je použití metody Monte Carlo proti algoritmům numerické matematiky vhodnější buď z hlediska rychlosti výpočtu nebo jednoduchosti algoritmu, případně z obou hledisek současně. Kritériem výběru témat jsou metody důležité při aplikaci ve fyzice.

4.4.1 Výpočet určitých integrálů

Základní metody

Úkolem bude vypočítat integrál z funkce $f(x)$ na vlastním intervalu $\langle a, b\rangle$

$$I = \int_a^b f(x)\,\mathrm{d}x\,. \tag{4.33}$$

K dispozici máme dvě jednoduché metody:

1. Výpočet střední hodnoty funkce:

   Principem metody je, že se nejprve pokusíme nalézt střední hodnotu integrované funkce $f(x)$ na intervalu $\langle a, b\rangle$, $\bar{f}$. Pokud se nám to podaří, hodnotu integrálu I určíme velmi snadno

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

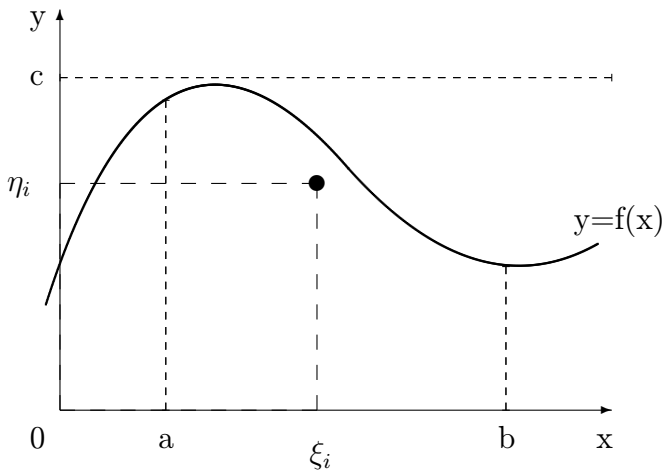
K nalezení střední hodnoty $\bar{f}$ použijeme metodu Monte Carlo. Princip spočívá v tom, že vezmeme náhodnou veličinu ξ rovnoměrně rozdělenou na intervalu $\langle a, b\rangle$ a zavedeme náhodnou veličinu $\eta = f(\xi)$, jejíž matematické očekávání $E\eta$ je rovno průměrné hodnotě funkce $f(x)$ na intervalu $\langle a, b\rangle$. Hodnotu $E\eta$ odhadneme pomocí zákona velkých čísel (4.16), tj. pomocí aritmetického průměru n nezávislých realizací veličiny η. Potom

$$I \doteq (b-a) \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} f(\xi_i)\,. \tag{4.34}$$

2. Geometrická metoda:

   V této metodě se vychází z geometrického významu integrálu jako plochy pod křivkou $y = f(x)$ v rozmezí a až b (obr. 4.4). Budeme předpokládat, že funkce $f(x)$ je na celém intervalu $\langle a, b\rangle$ omezená. Vezmeme dvě náhodné veličiny: $\xi \in \langle a, b\rangle$ a $\eta \in \langle 0, c\rangle$ rovnoměrně rozdělené na příslušných intervalech. Tímto postupem připravíme n náhodných bodů (ξ, η) a pomocí podmínky $\eta_i < f(\xi_i)$ budeme určovat, kolik z nich padne pod křivku $y = f(x)$. Jejich počet označíme n'. Potom za hodnotu integrálu I vezmeme odhad

$$I \doteq c\,(b-a) \cdot \frac{n'}{n}\,. \tag{4.35}$$



Obrázek 4.4: Geometrická metoda pro výpočet určitého integrálu funkce $f(x)$

Srovnání obou metod ukazuje, že metoda (4.35) má větší (nebo v optimálním případě stejný) rozptyl než metoda (4.34) založená na výpočtu střední hodnoty integrované funkce a bývá proto většinou méně

přesná. Z tohoto tvrzení ale automaticky nevyplývá, že metoda (4.34) je také efektivnější. Podle vztahu (4.31) metoda s větším rozptylem má i větší chybu a potřebuje pro dosažení stejné přesnosti větší počet pokusů. O efektivitě metody však rozhoduje celková doba potřebná k dosažení dané přesnosti a zde se musí vzít v úvahu délka výpočtu jedné realizace. Při porovnání počtu operací potřebných k dosažení jedné hodnoty pro výpočet veličiny I sice opět bude tento počet menší pro metodu střední hodnoty funkce než pro metodu geometrickou, délka trvání těchto operací však záleží na integrované funkci. Zatímco při realizaci jedné hodnoty v metodě střední hodnoty funkce je nutno počítat funkční hodnotu $f(\xi_i)$, při geometrické metodě je třeba pouze provádět test, zda nagenerovaný bod leží pod křivkou $y = f(x)$. Tento test má tvar $\eta_i < f(\xi_i)$ a jeho vhodnou úpravou se lze v případě potřeby výpočtu nevhodně složité funkce $f(x)$ vyhnout, což nelze udělat v případě metody střední hodnoty. Na obecné úrovni proto nelze o efektivitě obou metod rozhodnout.

Metody se zvýšenou účinností

Zatímco není možno bez znalosti konkrétní integrované funkce rozhodnout, zda je efektivnější metoda střední hodnoty nebo metoda geometrická, porovnání algoritmů založených na metodě Monte Carlo a klasických numerických algoritmů vyznívá jednoznačně ve prospěch postupů numerické matematiky. Z tohoto důvodu je nutno pro praktické použití obě výše uvedené metody zdokonalit. K dispozici máme celou řadu postupů a záleží na konkrétní integrované funkci, který z nich zvolit. Všechny tyto postupy nepředstavují algoritmy konkurenční k uvedeným základním metodám, ale jejich rozšíření, které je možno použít pro obě základní metody. Jelikož konvergence metody Monte Carlo je dosti pomalá (úměrná $1/\sqrt{n}$), je nutno přesnost výpočtu zvyšovat ne zvětšováním počtu pokusů n, ale zmenšováním rozptylu $D\xi$ – viz vztah pro chybu metody Monte Carlo (4.31). Uveďme si několik příkladů těchto postupů:

1. Nalezení hlavní části:

   Integrál se budeme snažit rozdělit na dvě části – jednu dávající hlavní vklad do výsledku a druhou jako upřesňující korekci $f(x) = g(x)+h(x)$:

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

   Pokud dokážeme první integrál z funkce $g(x)$ spočítat analyticky, budeme muset numericky určit jen hodnotu integrálu z funkce $h(x)$. Je-li však v prvním integrálu obsažena převážná část výsledku, projeví se

chyba v přibližném určení hodnoty druhého integrálu na výsledku jen málo a my můžeme buď získat celkový výsledek přesněji a nebo zmenšit počet pokusů n.

Jak již bylo řečeno, tento postup lze použít spolu jak s metodou střední hodnoty funkce tak i s metodou geometrickou. Např. pro metodu střední hodnoty se vztah (4.34) modifikuje na

$$I \doteq \frac{b-a}{n} \sum_{i=1}^{n} [f(\xi_i) - g(\xi_i)] + \int_a^b g(x)\,\mathrm{d}x\,.$$

Rozptyl nové náhodné veličiny, s kterou pracujeme,

$$\eta = (b-a)\,[f(\xi) - g(\xi)] + \int_a^b g(x)\,\mathrm{d}x\,,$$

bude zmenšený

$$D\eta = (b-a) \int_a^b [f(x) - g(x)]^2\,\mathrm{d}x \,-\, \left[I \,-\, \int_a^b g(x)\,\mathrm{d}x\right]^2$$

a efektivita metody střední hodnoty funkce vzroste.

2. Metoda váženého výběru:

   Při této modifikaci obou základních metod nebudeme volit náhodná čísla ξ v intervalu $\langle a, b\rangle$ rozdělená rovnoměrně, ale v oblasti více přispívající k hodnotě integrálu I je budeme generovat s větší hustotou. Tento postup existuje ve dvou modifikacích.

   V té jednodušší rozdělíme integrační interval $\langle a, b\rangle$ na několik dílčích intervalů, např. $\langle a, c_1\rangle$, $\langle c_1, c_2\rangle$,...,$\langle c_k, b\rangle$ a v každém budeme generovat jiný počet náhodných čísel ξ_i: n_1, n_2,...,n_{k+1}. Pokud zvolíme počty náhodných čísel n_i úměrné jak šířce dílčího intervalu tak i průměrné velikosti integrované funkce na tomto podintervalu, efektivita výpočtu se zvýší. Zvolíme-li $n_1 + n_2 + \cdots + n_{k+1} = n$ a použijeme-li pro integraci vztah

$$I = \int_a^{c_1} f(x)\,\mathrm{d}x \,+\, \int_{c_1}^{c_2} f(x)\,\mathrm{d}x \,+ \cdots + \int_{c_k}^{b} f(x)\,\mathrm{d}x\,,$$

   dostaneme výsledek přesnější než při práci s n náhodnými čísly ξ na celkovém intervalu $\langle a, b\rangle$ a to tím více, čím více se bude integrovaná funkce $f(x)$ na tomto intervalu měnit (a samozřejmě, čím lépe se dokážeme volbou dílčích intervalů a počty bodů v nich průběhu této funkce přizpůsobit).

Při obecnější a přesnější variantě metody váženého výběru nepracujeme v dílčích intervalech s konstantními hustotami bodů, ale budeme náhodnou veličinu ξ generovat s hustotou pravděpodobnosti spojitě se měnící na celém intervalu $\langle a, b\rangle$. Hledaný integrál upravíme do tvaru

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

Zavedeme-li novou náhodnou veličinu $\eta = f(\xi)/p(\xi)$, bude pro ni platit: $E\eta = I$. Jako odhad hledaného integrálu proto můžeme vzít vztah

$$I \doteq \frac{1}{n}\sum_{i=1}^{n}\frac{f(\xi_i)}{p(\xi_i)}\,.$$

Vhodnou volbou hustoty pravděpodobnosti $p(x)$ nyní můžeme podstatně zmenšit rozptyl a tím i chybu metody. Rozptyl náhodné veličiny η bude

$$D\eta = \int_a^b \frac{f(x)^2}{p(x)}\,\mathrm{d}x\; - I^2\,.$$

Pokud nyní zvolíme hustotu pravděpodobnosti veličiny ξ, $p(x)$, co nejpodobnější průběhu integrované funkce $f(x)$ (resp. její absolutní hodnoty), bude postup mnohem efektivnější než při předchozí variantě pracující s dílčími intervaly.

3. Metoda symetrizace integrované funkce:

   Obecně platí, že rozptyl metody bude tím menší, čím se bude integrovaná funkce na intervalu $\langle a, b\rangle$ pomaleji měnit – rozptyl konstanty je nula, resp. při určení střední hodnoty integrované funkce nám stačí jediný bod. Budeme se proto snažit integrovanou funkci upravit tak, aby rozptyl výpočtu poklesl, tj. aby se funkce na integračním intervalu měnila co nejméně. Konkrétní úprava bude záviset na tvaru integrované funkce $f(x)$.

   Je-li funkce $f(x)$ např. monotónně rostoucí nebo klesající, je ji vhodné symetrizovat podle vztahu

$$g(x) = \frac{1}{2}\left[f(x) + f(a+b-x)\right].$$

   Integrál pak budeme moci přibližně spočítat např. metodou střední hodnoty (4.34) podle modifikovaného vztahu

$$I \doteq \frac{b-a}{2n}\sum_{i=1}^{n}\left[f(\xi_i) + f(a+b-\xi_i)\right],$$

který bude určitě přesnější než původní vzorec. Podobné úpravy lze navrhnout i pro jiné průběhy integrované funkce.

Pokud nyní srovnáme postupy, které nám pro numerickou integraci funkce jedné proměnné nabízejí metoda Monte Carlo a numerická matematika, zjistíme, že postupy pro zvyšování účinnosti sice použitelnost metody Monte Carlo poněkud zvýšily, ve většině případů však ne dostatečně. Účinnost různých algoritmů především závisí na integrované funkci, obecně však lze říci, že při využití vhodných doplňujících postupů jak metoda střední hodnoty funkce tak i geometrická metoda budou pravděpodobně výhodnější než Newtonovy-Cotesovy vzorce, vztahy založené na Gaussově kvadratuře však budou ještě účinnější.

Ve prospěch metody Monte Carlo mluví pouze to, že její algoritmy jsou z programátorského hlediska velmi jednoduché. Na druhou stranu, nic nám nebrání v tom, abychom postupy navržené pro zvýšení účinnosti integrace metodou Monte Carlo nevyužili i v klasické numerické matematice a tím ještě více relativní účinnost algoritmů metody Monte Carlo nesnížili.

Vícerozměrné integrály

Zcela jiná situace však nastane, pokud změníme formulaci problému a místo integrování funkce jedné proměnné (4.33) budeme potřebovat počítat dvou- a vícerozměrné integrály

$$I = \int_a^b \int_{c(x)}^{d(x)} f(x, y)\, \mathrm{d}y\, \mathrm{d}x\,. \tag{4.36}$$

Tato situace je ve fyzice běžná, mnohem častěji než jednorozměrné případy řešíme případy dvourozměrné a třírozměrné. V některých oblastech fyziky jsou dokonce obvyklé postupy, při kterých je třeba integrovat přes celý fázový prostor, tj. řešit integrály šestirozměrné.

Zatímco pro jednorozměrné integrály existuje v numerické matematice větší množství výhodných integračních metod, přechod k vícerozměrným integrálům vždy znamená značné zkomplikování kvadraturních vzorců a velké prodloužení doby výpočtu. Budeme-li při výpočtu jednorozměrného integrálu brát funkční hodnoty v n uzlových bodech, při přechodu k d-rozměrnému problému vzroste počet uzlových bodů na n^d a známé metody numerické matematiky se stanou mimořádně neefektivní a při více rozměrech zcela nepoužitelné.

Naproti tomu metoda Monte Carlo toto omezení nemá. Obě základní metody navržené pro jednorozměrnou integraci i většinu doplňujících postupů zvyšujících efektivitu výpočtu lze bez problémů zobecnit na více rozměrů bez výrazné ztráty jejich účinnosti.

Obecná formulace problému (4.33) bude

$$I = \int_G f(P)\, p(P)\, \mathrm{d}P\,, \tag{4.37}$$

kde $p(P)$ je zadaná hustota pravděpodobnosti v oblasti G, přes kterou integrujeme, a $P = (x, y, \ldots)$ je bod z oblasti G.

Metoda střední hodnoty funkce spočívá v tom, že vezmeme náhodné body $P_1, P_2, \ldots$ s hustotou $p(P)$ (nejjednodušší případ je $p(P) = 1/G$) a zavedeme náhodnou veličinu $\eta = f(P)$, jejíž matematická naděje je rovna I

$$E\eta = \int_G f(P)\, p(P)\, \mathrm{d}P = I\,.$$

K nalezení odhadu hledaného integrálu můžeme použít vztah

$$\overline{\eta} = \frac{1}{n} \sum_{i=1}^{n} \eta_i = \frac{1}{n} \sum_{i=1}^{n} f(P_i)\,. \tag{4.38}$$

Analogicky můžeme zobecnit i geometrickou metodu. Budeme předpokládat, že funkce $f(P)$ je v oblasti G omezená, tj. že platí $0 \le f(P) \le c$. Vytvoříme si novou $d+1$ rozměrnou oblast $G \times (0, c)$ a v ní budeme generovat náhodné body Q. Budeme vyšetřovat, kolik z těchto n bodů bude ležet pod povrchem „plochy“ $z = f(P)$. Nechť jejich počet je n'. Pro integrál I, tj. pro $d+1$ rozměrný „objem“, pak dostaneme odhad

$$I \doteq c\, G\, \frac{n'}{n}\,, \tag{4.39}$$

srovnej jednorozměrný vztah (4.35).

Ve více rozměrech lze též používat postupy pro snižování rozptylu, i když to někdy bude náročnější. Zejména symetrizace integrované funkce může pro oblast G složitějšího průběhu činit obtíže. Proti jednorozměrnému případu však přibývá další pravidlo: Dokážeme-li integrovat přes některou proměnnou analyticky, rozptyl se sníží.

Použití metody Monte Carlo ve více rozměrech je pro hladkou integrovanou funkci $f(x)$ a vhodnou oblast integrace G výhodnější než klasické kvadraturní vzorce pro $d \ge 3$. Bude-li funkce $f(x)$ spojitá jen po částech a nebo bude-li složitá oblast G, používá se metoda Monte Carlo již pro $d \ge 2$. Pouze pro mimořádně složité průběhy funkce $f(x)$ může být výhodné použít metodu Monte Carlo i v jednorozměrném případě.

4.4.2 Řešení Laplaceovy rovnice

Další oblastí matematiky, která má své podstatné použití v moderní fyzice, je řešení parciálních diferenciálních rovnic. Z těchto rovnic se v různých oblastech fyziky velmi často vyskytují Laplaceova a Poissonova rovnice, ať již při výpočtu rozložení elektrostatického potenciálu, vedení tepla v látkách, řešení vlnové rovnice, proudění plynů, modelování plazmatu, řešení Schrödingerovy rovnice, atd. V tomto studijním textu budeme demonstrovat použití metody Monte Carlo při řešení eliptických parciálních diferenciálních rovnic na řešení Laplaceovy rovnice s Dirichletovou úlohou, kdy hledáme řešení Laplaceovy rovnice v určité oblasti a máme k dispozici hodnoty hledané funkce na hranici této oblasti.

Formulace úlohy

Z terminologických důvodů si budeme problém formulovat pro případ, kdy máme stanovit rozdělení elektrostatického potenciálu ve dvou rozměrech:

- Vezměme si dvourozměrnou oblast G s hranicí Γ.
- Mějme za úkol nalézt rozložení potenciálu $U(x, y)$ vyhovující uvnitř oblasti G Laplaceově rovnici $\Delta U = 0$ a nabývající na hranici oblasti hodnoty $U(\Gamma)$.
- Zavedeme čtvercovou síť s krokem h očíslovanou indexy i a j (ve vodorovném a svislém směru). Provedeme diskretizaci prostorových souřadnic, tj. místo spojitě rozložených bodů o souřadnicích (x, y) budeme pracovat pouze s uzly sítě o souřadnicích (i, j).
- Místo se spojitým potenciálem $U(x, y)$ budeme pracovat s novou diskrétní funkcí u_{ij}.
- Postupem obvyklým z numerické matematiky převedeme studovanou parciální diferenciální rovnici na její diferenční ekvivalent, tj. nejprve nalezneme ekvivalenty parciálních derivací $\partial U/\partial x$, $\partial U/\partial y$, $\partial^2 U/\partial x^2$, atd. a z nich sestavíme diferenční ekvivalent celé rovnice. Pro Laplaceovu rovnici dostaneme obvyklý čtyřbodový vztah

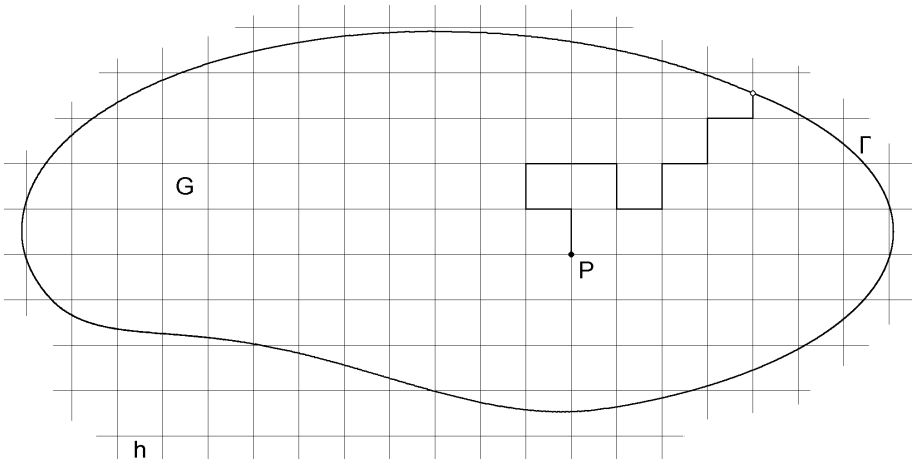
$$u_{ij} = \frac{1}{4}\left[u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}\right]. \tag{4.40}$$

- Tento vztah musí platit současně ve všech vnitřních bodech oblasti G a při řešení se musí využít i hodnoty diskretizované funkce u na hranicích oblasti Γ, kde pro uzlové body hranice prostě položíme $U(\Gamma) = u(\Gamma)$.

Vidíme, že formulace úlohy je shodná s přístupem numerické matematiky. Rozdíl bude v řešení vzniklé soustavy rovnic (4.40) – místo superrelaxační metody, Gaussovy eliminace a dalších numerických algoritmů se využijí stochastické algoritmy metody Monte Carlo. Uvedenou formulaci problému můžeme modifikovat podle konkrétních podmínek – čtvercovou síť nahradit obdélníkovou či jiného tvaru, diferenční rovnici sestavit pro jiný typ parciální diferenciální rovnice, místo dvoudimenzionální úlohy pracovat ve třech rozměrech, atd.

Algoritmus bloudění v pravoúhlé síti

Jednoduchý algoritmus pro řešení Laplaceovy rovnice je založený na představě náhodného bloudění ve vytvořené síti - viz obr. 4.5. Budeme hledat pravděpodobnost toho, že po vypuštění z bodu P dorazíme po náhodné trajektorii do některého hraničního bodu, kde již zůstaneme. S každým hraničním bodem je spojena nějaká číselná veličina, kterou při tomto bloudění získáme. Celý pokus pak n-krát zopakujeme a startovnímu bodu P přiřadíme průměrnou hodnotu z takto získaných veličin. Podstata popisovaného algoritmu spočívá v tom, že pokud jako číselné veličiny v hraničních bodech vezmeme hodnoty $U(\Gamma)$, dostaneme tímto postupem přibližnou hodnotu potenciálu v bodě P, $U(P)$.



Obrázek 4.5: Síť pro řešení dvourozměrné Laplaceovy rovnice metodou Monte Carlo. Algoritmus bloudění v pravoúhlé síti.

Při náhodném bloudění musíme procházet řadou dalších bodů („křižovatek“) a v každém z nich máme stejnou pravděpodobnost, že budeme pokračovat v jednom ze čtyř možných směrů. Pokud tuto podmínku vyjádříme v

pravděpodobnostech, dostaneme výraz ekvivalentní vztahu (4.40), jenž odpovídá numerickému řešení právě Laplaceovy rovnice.

Popsaný algoritmus má výrazné výhody i nevýhody. Velkou výhodou je, že proti numerickému řešení Laplaceovy rovnice, kdy musí vztah (4.40) současně platit ve všech vnitřních bodech oblasti G, tj. kdy musíme řešit soustavu velkého počtu algebraických rovnic, zde získáme řešení v jednom bodě P. Pokud proto je naším cílem nalézt řešení Laplaceovy rovnice v jediném bodě nebo v malém počtu bodů oblasti G, může být výpočet metodou Monte Carlo výhodnější než obvyklé numerické algoritmy i v této jednoduché podobě. Pokud však potřebujeme nalézt řešení Laplaceovy rovnice současně ve všech vnitřních bodech oblasti G, mnohem efektivnější postup nám nabízí klasická numerická matematika.

Základní slabinou algoritmu bloudění v pravoúhlé síti je právě ta síť. Používáme ji ke dvěma účelům, které jsou vůči sobě v protikladu:

- Síť nám slouží k tomu, abychom v jejích uzlech určovali hledané hodnoty potenciálu. Pro dosažení dostatečné přesnosti by proto síť měla být co nejhustší.

- Síť slouží též k tomu, abychom s její pomocí realizovali co nejnáhodnější procházku. V tomto případě příliš hustá síť již náhodnost nezvýší, pouze prodlouží dobu výpočtu.

Existují algoritmy, které obě funkce sítě od sebe oddělují a celý výpočet tím zefektivňují. Příklad takového algoritmu je znázorněn na obr. 4.6. Tento postup je výrazně silnější, ale jeho aplikace je též o dost složitější, proto jej lze doporučit jen zkušeným uživatelům a nebo v případě oblasti G s hranicí velmi jednoduchého tvaru.

Algoritmus bloudění s náhodným krokem

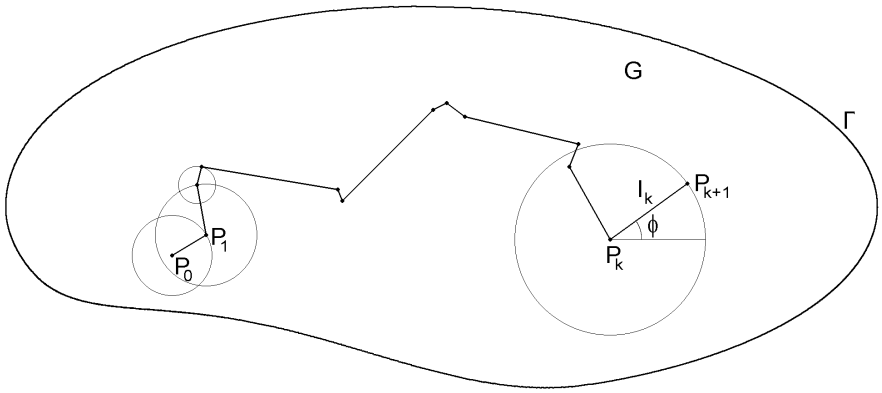
Základní nevýhodou algoritmu bloudění ve čtvercové síti je malá délka každého kroku h. Následující algoritmus proto rozšiřuje náhodnost i na volbu délky kroku, což má za následek rychlejší konvergenci:

Trajektorii sestrojíme podle obr. 4.6. Vyjdeme z bodu P_0. Vytvoříme kružnici o poloměru l_0 na nějž klademe jediné omezení, že celá kružnice musí ležet v oblasti G, jinak je velikost poloměru náhodná. Na této kružnici náhodně zvolíme bod P_1 a celý postup opakujeme. Dostaneme tak náhodnou trajektorii tvořenou body $P_0, P_1, P_2, \ldots$. Obecně bude platit

$$P_{k+1} = P_k + l_k\,\omega_k \quad k = 0, 1, 2, \ldots\ ,$$

kde $\omega_k = \cos\phi_k + \sin\phi_k$ a náhodný úhel ϕ_k je rovnoměrně rozdělen v intervalu $\langle 0, 2\pi\rangle$.

Při implementaci tohoto algoritmu však musíme vyřešit dva dosti složité problémy. Prvním je ukončení trajektorie. Zatímco v případě bloudění v pravoúhlé síti trajektorie končí při dosažení hranice Γ (ať již v její skutečné poloze nebo v nejbližším uzlovém bodě), algoritmus bloudění s náhodným krokem by prakticky nikdy nekončil - hranice bychom teoreticky mohli dosáhnout pouze v případě, kdy nagenerujeme kružnici o maximálním možném poloměru a současně rozehrajeme úhel směřující kolmo k této hranici. Proto je nutno pojem hranice nově definovat jako pás šířky ε. Kritérium ukončení trajektorie pak změníme tak, že trajektorie končí v okamžiku vstupu do tohoto hraničního pásu. Druhá obtíž spočívá ve volbě kritéria pro generování poloměru kružnice l. Tento poloměr volíme v každém kroku trajektorie jako rovnoměrně rozdělenou náhodnou veličinu ležící v mezích $(0, i_{max}\rangle$ a právě stanovení maximálního poloměru l_{max} je v případě složitějšího tvaru hranice obtížné a zpomaluje výpočet.



Obrázek 4.6: Konstrukce trajektorie pro řešení Laplaceovy rovnice při použití algoritmu bloudění s náhodným krokem.

Vedle Dirichletovy úlohy, kde máme jako vstupní hodnoty zadány hodnoty $U(\Gamma)$ na hranici Γ, existují i úlohy, kdy v některých nebo všech bodech hranice máme zadány hodnoty intenzity elektrického pole, tj. derivace veličiny U. I takové úlohy lze řešit modifikovanými algoritmy náhodného bloudění, kdy připustíme po doražení do hraničního uzlu s určitou pravděpodobností další bloudění v síti, a nebo naopak, kdy bloudění ukončíme dříve, než dorazíme na hranici. Další možné modifikace algoritmů náhodného bloudění jsou použitelné i pro jiné typy parciálních diferenciálních rovnic, kdy změněné podmínce typu (4.40) přiřadíme odpovídajícím způsobem změněné migrační podmínky na „křižovatkách“.

Účinnost metody Monte Carlo při řešení Laplaceovy rovnice

Podobně jako u numerické integrace, použití metody Monte Carlo na řešení Laplaceovy rovnice přináší sice jednoduchý algoritmus, ale jeho efektivita není příliš vysoká. Situace se však opět výrazně mění s přechodem do vyšších dimenzí. V klasických numerických algoritmech, přechod z dvourozměrné do třírozměrné úlohy znamená značné prodloužení výpočtu. Typické minimální rozměry sítí při praktickém použití numerického řešení Laplaceovy rovnice např. v elektrotechnice jsou řádu 100×100, tj. 10^4. Přechod na třírozměrnou síť znamená řešit řádově 10^6 algebraických rovnic typu (4.40). Naproti tomu v algoritmech metody Monte Carlo stoupá výpočetní náročnost problému s dimenzí mnohem pomaleji. Označíme-li dobu potřebnou k přechodu z jednoho bodu náhodné trajektorie do dalšího jako τ a celkový počet kroků jedné trajektorie při i-tém pokusu jako ν_i, bude celková doba výpočtu

$$T = \tau \left(\nu_1 + \nu_2 + \cdots + \nu_n\right) = \tau \, n \, E\nu \, .$$

Počet pokusů n potřebných k dosažení zadané přesnosti najdeme ze zákona velkých čísel. Celková doba výpočtu T závisí na lineárních rozměrech oblasti G kvadraticky a při přechodu do třetí dimenze se tento odhad změní jen málo.

Další výhodou použití metody Monte Carlo pro řešení Laplaceovy rovnice jsou malé nároky na paměť počítače. Zatímco v síťových algoritmech numerické matematiky je třeba udržovat v paměti počítače současně celou síť, tj. minimálně 10^4 až 10^6 proměnných (podle dimenzionality problému), není u algoritmů náhodného bloudění nutné zaznamenávat celou síť ani trajektorii, stačí pouze znát souřadnice počátečního bodu a okamžitou polohu.

Podstatnou výhodou algoritmů založených na metodě Monte Carlo je, že není nutné současně počítat rozdělení potenciálu ve všech uzlech sítě, ale jen ve vybraných bodech. Pokud to řešení našeho problému dovoluje nebo přímo vyžaduje, bude použití metody Monte Carlo výhodné již pro úlohy dvourozměrné, v obecném případě pak začne být metoda Monte Carlo výhodnější při řešení úloh prostorových.

4.4.3 Další problémy

Vedle dvou oblastí numerické matematiky, kde se za určitých podmínek výrazně uplatňuje metoda Monte Carlo a které jsme si probrali podrobněji, lze konstatovat, že stochastické algoritmy lze nalézt téměř pro každou oblast numerických výpočtů a otázkou pouze je, kdy je jejich použití výhodné. Metoda Monte Carlo se s úspěchem používá ve fyzice vedle numerické integrace a ře-

šení parciálních diferenciálních rovnic, zejména při interpolaci funkcí mnoha proměnných a při řešení soustav lineárních algebraických rovnic.

Při interpolaci funkcí mnoha proměnných se využívá efektivnost stochastických algoritmů při přechodu k vyšším dimenzím, takže výhodnost metody Monte Carlo se projeví tím více, čím více proměnných interpolovaná funkce obsahuje. Při řešení soustav lineárních algebraických rovnic se s úspěchem využívá jiná schopnost stochastických algoritmů – soustředit se na řešení jen omezeného problému, což v tomto případě znamená hledání jednoho konkrétního kořene. Zatímco běžné metody současně nacházejí všechny kořeny, což vede na problém složitosti m^2, kde m je počet rovnic soustavy, metoda Monte Carlo dokáže hledat pouze jeden kořen čímž se doba výpočtu redukuje na veličinu úměrnou m.

Mezi velmi efektivní aplikace metody Monte Carlo patří ***interpolace funkcí mnoha proměnných***. Mějme funkci m proměnných, $f(x_1, x_2, \ldots, x_m)$, a mějme zadány její hodnoty pouze ve vrcholech jednotkové m-rozměrné krychle. Naším úkolem je určit hodnotu funkce v nějakém bodě o souřadnicích $(x_1, \ldots, x_m)$ uvnitř krychle, přičemž podle každé souřadnice budeme interpolovat lineárně.

Interpolační vztahy pro jednotlivé rozměry m budou

$$\begin{aligned}
m = 1 \qquad f(x_1) &= (1 - x_1) \cdot f(0) + x_1 \cdot f(1) \\
m = 2 \quad f(x_1, x_2) &= (1 - x_1) \cdot (1 - x_2) \cdot f(0,0) + \\
&\quad + x_1 \cdot (1 - x_2) \cdot f(1,0) + \\
&\quad + (1 - x_1) \cdot x_2 \cdot f(0,1) + x_1 \cdot x_2 \cdot f(1,1)\,.
\end{aligned}$$

S růstem počtu proměnných m počet členů v interpolačních vztazích velmi rychle narůstá – např. pro $m = 30$ dosahuje 10^9.

Naproti tomu algoritmus řešení téže úlohy metodou Monte Carlo je velmi jednoduchý. Vytvoříme n bodů o náhodných souřadnicích $(\xi_1, \ldots, \xi_m)$ a z nich určíme hodnotu interpolované funkce f

$$f(x_1, \ldots, x_m) \doteq \frac{1}{n} \sum_{i=1}^{n} f(\xi_1^{(i)}, \ldots, \xi_m^{(i)})\,.$$

Každá z veličin $\xi_j^{(i)} (j = 1, \ldots, m;\ i = 1, \ldots, n)$ může nabývat hodnot 0 nebo 1 s pravděpodobnostmi

$$\begin{aligned}
P\{\xi_j = 0\} &= 1 - x_j \\
P\{\xi_j = 1\} &= x_j\,,
\end{aligned}$$

proto každý bod $(\xi_1^{(i)}, \ldots, \xi_m^{(i)})$ představuje některý vrchol jednotkové krychle. Hodnoty ξ_j určíme pomocí náhodné veličiny γ rovnoměrně

rozdělené v intervalu $(0,1)$ podle předpisu

$$\begin{aligned} \xi_j &= 0 \quad \text{pro} \quad \gamma \geq x_j \\ \xi_j &= 1 \quad \text{pro} \quad \gamma < x_j \,. \end{aligned}$$

Dalším problémem, kde se za určitých podmínek může uplatnit i metoda Monte Carlo, je ***řešení soustav lineárních algebraických rovnic***. Numerická matematika nabízí řadu postupů, např. dále uvedený algoritmus.

Zformulujme si úlohu: Vezměme soustavu m lineárních algebraických rovnic s neznámými $z_1, \ldots, z_m$

$$\sum_{j=1}^{m} a_{ij}\, z_j = b_i \qquad i = 1, \ldots, m \,. \tag{4.41}$$

Tuto soustavu můžeme zapsat ve vektorové formě jako

$$A\,\vec{z} = \vec{b}\,,$$

kde A je čtvercová matice typu $m \times m$, $\vec{z}$ a $\vec{b}$ jsou vektory o m složkách. Ve většině případů je možno tuto soustavu přepsat do „iteračního“ tvaru

$$\vec{z} = C\,\vec{z} + \vec{b}\,,$$

kde matice C je rovna $E - A$, E je jednotková matice. Řešení má tvar

$$\vec{z} = A^{-1}\,\vec{b}\,. \tag{4.42}$$

Inverzní matici A^{-1} můžeme vyjádřit řadou

$$A^{-1} = E + C + C^2 + \cdots \,.$$

Řada konverguje tehdy, jsou-li vlastní čísla matice C v absolutní hodnotě menší než 1. Dosadíme-li rozvoj matice A^{-1} do hledaného řešení, dostaneme

$$\vec{z} = \vec{b} + C\,\vec{b} + C^2\,\vec{b} + \cdots \,.$$

Toto řešení můžeme složit z dílčích iterací

$$\begin{aligned} \vec{z}^{(1)} &= \vec{b} \\ \vec{z}^{(2)} &= C\,\vec{z}^{(1)} + \vec{b} = C\,\vec{b} + \vec{b} \\ \vec{z}^{(3)} &= C\,\vec{z}^{(2)} + \vec{b} = C^2\,\vec{b} + C\,\vec{b} + \vec{b} \\ &\ldots \end{aligned}$$

Metoda jednoduchých iterací konverguje právě tehdy, konverguje-li řada $\vec{z}^{(1)}, \vec{z}^{(2)}, \vec{z}^{(3)}, \ldots$. Pro i-tou složku vektoru řešení, z_i, dostaneme vztah

$$\begin{aligned} z_i &= b_i + \sum_j c_{ij}\, b_j + \sum_{j_1} \sum_{j_2} c_{ij_1}\, c_{j_1 j_2}\, b_{j_2} + \\ &\quad + \sum_{j_1} \sum_{j_2} \sum_{j_3} c_{ij_1}\, c_{j_1 j_2}\, c_{j_2 j_3}\, b_{j_3} + \cdots . \end{aligned} \tag{4.43}$$

A nyní se podíváme na řešení téhož problému pomocí metody Monte Carlo. Vztah (4.43) nám udává návod, jak vypočítat jednu složku řešení rovnic (4.41). Jednotlivé členy výrazu (4.43) můžeme určit pomocí náhodných veličin. Naším cílem bude nalézt náhodnou veličinu η_i, jejíž matematická naděje je rovna součtu řady (4.43), tj. $E\eta_i = z_i$. Budeme-li postup n-krát opakovat, dostaneme jednotlivé realizace veličiny $\eta_i : \eta_i^{(1)}, \eta_i^{(2)}, \ldots, \eta_i^{(n)}$. Jako aproximaci řešení z_i pak vezmeme aritmetický průměr těchto veličin $\bar{\eta}_i$.

Veličinu η_i zkonstruujeme následujícím způsobem: Každý prvek matice C zapíšeme jako součin dvou prvků

$$c_{ij} = F_{ij}\, P_{ij}\,,$$

kde $P_{ij} \in \langle 0, 1)$; pro $c_{ij} = 0$ položíme $P_{ij} = 0$. Analogicky upravíme i prvky vektoru pravé strany $\vec{b}$

$$b_i = f_i\, p_i\,,$$

$p_i \in (0, 1)$. Volbu prvků P a p provedeme tak, aby platilo

$$p_i + \sum_{j=1}^{m} P_{ij} = 1 \qquad i = 1, \ldots, m\,.$$

Na tyto prvky odpovídající jedné rovnici soustavy (4.42) můžeme pohlížet jako na soubor pravděpodobností nezávislých událostí. Rovnici (4.43) s jejich pomocí přepíšeme do tvaru

$$\begin{aligned} z_i &= f_i\, p_i + \sum_j F_{ij}\, f_j\, P_{ij}\, p_j + \cdots + \\ &\quad + \sum_{j_1} \cdots \sum_{j_r} F_{ij_1} \cdots F_{j_{r-1} j_r}\, f_{j_r}\, P_{ij_1} \cdots P_{j_{r-1} j_r}\, p_{j_r} + \\ &\quad + \cdots \end{aligned} \tag{4.44}$$

Pro $i = 1$ nyní rozdělíme interval $\langle 0, 1\rangle$ na $m + 1$ částí, jejichž délky budou

$$P_{i1}, P_{i2}, \ldots, P_{im}, p_i\,.$$

Totéž můžeme provést i pro ostatní hodnoty $i = 2, \ldots, m$ a získat tak m způsobů rozdělení intervalu $\langle 0, 1\rangle$. Pak nagenerujeme náhodné veličiny nezávislé a rovnoměrně rozdělené na intervalu $(0, 1)$: $\gamma_0, \gamma_1, \gamma_2, \ldots$. Nejprve vezmeme i-té rozdělení intervalu $\langle 0, 1\rangle$ a zjistíme, kam padne veličina γ_0. Padne-li do poslední, $m + 1$. části délky p_i, položíme realizaci náhodné veličiny η_i rovnou f_i. Padne-li číslo γ_0 do j-té části rozdělení (o délce P_{ij_1}), vezmeme další, j_1-té rozdělení intervalu $\langle 0, 1\rangle$

$$P_{j_1 1}, P_{j_1 2}, \ldots, P_{j_1 m}, p_{j_1}$$

a zjistíme, do které části tohoto nového rozdělení padne číslo γ_1. V případě, že γ_1 padlo do poslední části délky p_{j_1}, vezmeme jako hodnotu veličiny η_i číslo $F_{ij_1} f_{j_1}$. Padlo-li γ_1 do j_2-té části rozdělení, postup zopakujeme s j_2-tým rozdělením intervalu $\langle 0, 1\rangle$ a číslem γ_2. Tento postup ukončíme, až číslo γ_r padne do poslední části příslušného rozdělení j_r. Pak za realizaci náhodné veličiny η_i v tomto experimentu vezmeme

$$\eta_i = F_{ij_1} F_{j_1 j_2} \cdots F_{j_{r-1} j_r} f_{j_r} .$$

Veličina η_i nabývá této hodnoty s pravděpodobností

$$P_{j_1 1}, P_{j_1 j_2}, \ldots, P_{j_1 j_r}, p_{j_r} ,$$

proto střední hodnota veličiny η_i je rovna vztahu (4.44). Opakováním celého procesu n-krát získáme veličiny $\eta_i^{(1)}, \ldots, \eta_i^{(n)}$, spočítáme aritmetický průměr a položíme jej přibližně roven z_i.

Při srovnání s klasickými metodami numerické matematiky ale zjistíme, že popsaný proces je příliš pomalý a pokud potřebujeme najít úplné řešení soustavy lineárních algebraických rovnic, je třeba používat standardní metody. Výpočet metodou Monte Carlo je vhodný jen tehdy, potřebujeme-li znát hodnotu jediného kořenu z_i (a bude-li soustava rovnic dosti veliká).

4.5 Použití metody Monte Carlo ve fyzice

Jak vyplynulo z předchozí sekce, metoda Monte Carlo může najít své místo v matematice a jejím prostřednictvím i ve fyzice. Hlavní fyzikální použití metody Monte Carlo je ovšem při provádění počítačových experimentů. V sekci 3.2 jsme se seznámili s různými technikami počítačového modelování a z ní vyplynulo, že typické použití metody Monte Carlo je při stochastickém částicovém modelování. Problematice stochastického částicového modelování ve fyzice se budeme věnovat v této sekci. Je však třeba úvodem poznamenat, že touto (i když nejvýznamnější) aplikací metody Monte Carlo její význam

pro počítačovou fyziku nekončí. Podobně jako při řešení problémů numerické matematiky lze i v počítačové fyzice nalézt možné aplikace stochastických postupů řešení v mnoha dalších oblastech vědy, otázkou pouze je, zda je to výhodnější než standardní postupy příslušné oblasti.

Při řešení problémů vyjádřených pomocí postupů numerické matematiky, ať již klasických nebo jejich stochastických ekvivalentů, jsme schopni popsat konkrétní algoritmus řešení studovaného problému, tento algoritmus naprogramovat a pro konkrétní data vyřešit. Na rozdíl od této situace při počítačových experimentech ve fyzice stojíme před podstatně složitějším úkolem – fyzikálních problémů je mimořádně velký počet a teoreticky každý z nich si vyžaduje individuální přístup. Je proto velmi obtížné na obecné úrovni hledat postup řešení a též jej vyložit ve studijním textu.

Pokud se vrátíme k obecnému schématu řešení problémů pomocí metody Monte Carlo (Sekce 3.1):

1. Analýza problému a stochastická formulace modelu pomocí náhodné veličiny ξ

2. Generování náhodné veličiny γ rovnoměrně rozdělené na jednotkovém intervalu

3. Transformace náhodné veličiny γ na hledanou náhodnou veličinu ξ

4. Statistické vyhodnocení opakováním bodů 2 a 3 vzniklého souboru realizací náhodné veličiny ξ: $\xi_1, \xi_2, \ldots, \xi_n$,

těžiště počítačového experimentu stochastickým modelováním je v kroku 1.

Při analýze studovaného fyzikálního problému je třeba nalézt vhodný zjednodušený model a určit, která náhodná veličina jej nejlépe popisuje. Do tohoto kroku též patří určení toho, kde najdeme hledaný výsledek – např. v prvním momentu této náhodné veličiny ξ nebo v tabulce pravděpodobností diskrétní náhodné veličiny (4.3) či hustotě pravděpodobnosti spojité náhodné veličiny (4.4). Momenty obvykle vyhodnocujeme tehdy, když hledáme fyzikální řešení ve formě jediné hodnoty, zatímco např. úhlová nebo energetická rozdělení částic nejčastěji převádíme na rozdělení pravděpodobností náhodných veličin. Tato analýza je velmi důležitá, neboť určuje techniku statistického vyhodnocení výsledků v bodě 4.

Vlastní tvůrčí činnost leží v bodě 1, který je nejobtížnější. Vše ostatní jsou již standardní postupy, které jdou naučit (a po dočtení studijního textu do tohoto místa by je již čtenář měl znát – pokud tomu tak není, je třeba si zopakovat látku obsaženou v sekcích 4.2 a 4.3).

4.5.1 Transportní problém

I když předchozí výklad byl pesimistický v tom, že vlastně každé řešení fyzikálního problému metodou Monte Carlo je originální a nelze se mu proto naučit, realita ve skutečnosti tak nevýhodná není.

Nejrozšířenějším problémem částicového modelování ve fyzice je tzv. transportní problém, při kterém studujeme průchod částic hmotným prostředím. Transportní problém lze konkrétním algoritmem (či lépe souborem algoritmů) popsat a na tuto formulaci lze převést velké množství nejrůznějších fyzikálních úloh. Původní formulace pochází z oblasti jaderné fyziky (mezi tvůrce metody Monte Carlo patřili významní jaderní fyzici 40. let), kde byl studován transport neutronů v atomovém reaktoru. Lze tak ale řešit např. také

- transport elektronů v polovodičích
- průchod urychlených elektronů v elektronovém mikroskopu tenkou vrstvou kovu
- pohyb elektronů a iontů v plazmatu (především horkém)
- průchod záření látkou, ...

Algoritmy pro všechny tyto problémy jsou společné a záleží pouze na fyzikální interpretaci vstupních dat a výsledků modelování.

Základní schéma

Pokud bychom řešili transport částic látkou jinými prostředky než metodou Monte Carlo, museli bychom detailně znát celý průběh transportu a ten ce nejlépe napodobit výpočetními prostředky – spojitým modelováním nebo deterministickým částicovým modelováním. Metoda Monte Carlo je z tohoto hlediska mnohem bližší reálnému experimentu. Před začátkem modelování je třeba (experimentálně nebo teoreticky) určit podstatné fyzikální procesy, ke kterým při transportu dochází, a jejich rozdělovací zákony. Při vlastní simulaci pak postupně rozehráváme jednotlivé hodnoty těchto náhodných faktorů a započtením jejich vlivu určíme konkrétní realizaci studovaného náhodného procesu.

Tímto postupem můžeme modelovat historii částice a následným zobecněním velkého počtu těchto historií určit i makroskopické charakteristiky procesu. Tak můžeme naši představu o reálném procesu srovnat s experimentem, experiment za pomoci výsledků modelování upravit a v mnoha případech jej dokonce nahradit počítačovým výpočtem. Proto se uvedené metodě také říká ***metoda počítačového experimentu***. Klasický přístup spojitého modelování naproti tomu po celou dobu pracuje pouze s makroskopickými

veličinami – na základě analýzy experimentálních dat zformulujeme rovnice pro tyto veličiny a snažíme se je řešit.

Při počítačovém experimentu můžeme postupovat dvěma cestami. Při první nejprve analyzujeme celý studovaný fyzikální jev a snažíme se vybudovat model co nejvěrnější, ale přitom dostatečně jednoduchý pro následující výpočet. Takovým modelům se říká ***přirozené*** a budeme se jimi zabývat v této sekci. Alternativnímu postupu pomocí tzv. ***umělých modelů*** se budeme věnovat v sekci 4.5.2.

Data

Při počítačovém modelování musíme spolu s výběrem používaných algoritmů pozornost věnovat i *datovým strukturám*, kam budou data v průběhu výpočtu ukládána. To vyplývá ze zásad strukturovaného programování – viz název základní knihy prof. Wirtha „Algorithms + Data Structures = Programs“ [2]. Datová struktura vhodná pro počítačové studium transportu částic je uvedena na obr. 4.7.

	1	2	3	4	5	...												
x																		
y																		
z																		
v_x																		
v_y																		
v_z																		
typ																		
...																		

Obrázek 4.7: Datová struktura pro částicové modelování – přirozené modely.

Všechny údaje příslušející jedné částici jsou ukládány do jednoho sloupce datové struktury, tj. počet sloupců odpovídá celkovému počtu částic.

> Ve skutečnosti je situace poněkud složitější. Při částicovém modelování rozeznáváme dva typy algoritmů. Odlišují se tím, kolik částic musí být přítomno v pracovní oblasti současně (v rámci modelu, ne ve skutečném jevu). Pokud je rozhodující pouze interakce částic s prostředím a vzájemnou interakci částic můžeme zanedbat, zjednodušíme

si řešení problému tím, že budeme při simulaci vypouštět částice postupně a teprve když urazí celou svou trajektorii, budeme sledovat pohyb další částice. Toto je tzv. *jednočásticová metoda*. Jí by odpovídala datová struktura na obr. 4.7 pouze s jedním sloupcem.

Pokud ale bude vedle interakce s prostředím přítomna i vzájemná interakce částic, případně pokud studujeme problém, kde částice interagují pouze spolu (což je případ průchodu nabitých částic vakuem, případně pohyb gravitačně interagujících těles opět ve vakuu), musíme současně sledovat chování celého souboru částic. Pak se bude jednat o *metodu mnohočásticovou* a při jejím použití musíme pracovat s velkou datovou strukturou.

Svou podstatou je metoda Monte Carlo jednočásticová, zatímco deterministická metoda molekulární dynamiky je mnohočásticová. V současné době ale má značná část algoritmů částicového modelování rysy jak deterministické tak i stochastické a proto je obvyklý mnohočásticový přístup i v metodě Monte Carlo.

Současné částicové výpočty pracují s velkým počtem částic, někdy i řádu 10^6 až 10^8 a pak velká datová struktura znamená ohrožení výpočtu, neboť může přesáhnout kapacitu interní paměti počítače. Je proto nezbytné data do této struktury ukládat co nejúsporněji.

Nyní si datovou strukturu uvedenou na obr. 4.7 prohlédněme detailně. Pro každou částici do ní budeme ukládat několik skupin údajů:

– Prostorové souřadnice částice
  Naše modely mohou být prostorově jednorozměrné nebo vícerozměrné, takže první skupina dat bude obsahovat 1-3 položky, např. kartézské souřadnice x, y a z (jako v našem příkladu), nebo souřadnice v soustavě sférické, atd. V každém případě ale tato skupina nemůže být prázdná, neboť musí obsahovat minimálně jednu prostorovou souřadnici, která při studiu průchodu částic látkou představuje jakousi řídící osu (a jíž odpovídá hlavní cyklus v programu). V algoritmech metody molekulární dynamiky má podobné postavení osa časová.

  To, co bylo zde napsáno, platí pro klasickou formulaci metody Monte Carlo. Existují však i moderní postupy – především tzv. kinetická metoda Monte Carlo – která se v tomto ohledu podobá metodě molekulární dynamiky a má jako hlavní proměnnou též čas.

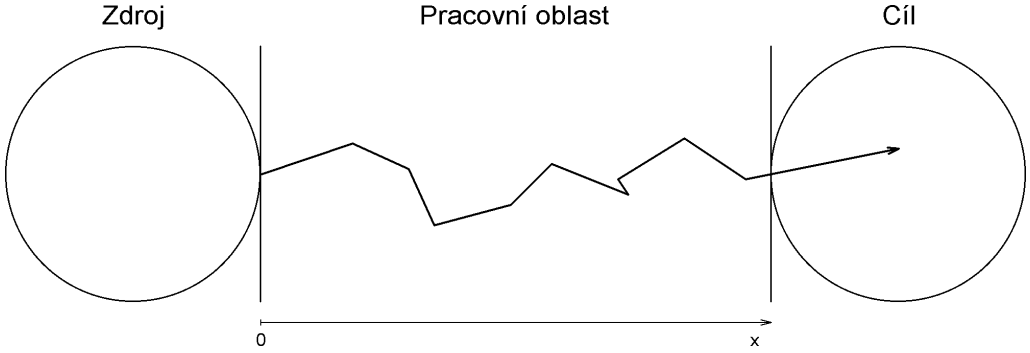
– Rychlosti částic
  Abychom mohli sledovat průběh transportu, měli bychom mít k dispozici informace o okamžité rychlosti každé částice $\vec{v}$. Opět v závislosti na rozměrnosti modelu se může jednat např. o složky rychlosti v_x, v_y a v_z, případně vyjádřené v jiné soustavě, ale také např. o absolutní hodnotu

celkové rychlosti v spolu se složkou v_x, atd. V některých aplikacích je vhodnější pracovat s kinetickou energií částice, tj. s hodnotami E_x, E_y a E_z, a opět v různých kombinacích – např. E a E_x. Obecně platí, že je vhodné si datovou strukturu co nejlépe přizpůsobit řešenému problému tak, aby byla data ukládána co nejúsporněji a též, aby se s nimi manipulovalo co nejrychleji – to platí zejména při volbě formátu údajů buď ve tvaru rychlostním nebo energetickém.

– Další údaje
Při řešení řady problémů výše uvedené dvě skupiny údajů o částici stačí. Pokud však ale budeme mít v našem souboru více druhů částic, což je typické např. pro fyziku plazmatu nebo pro jadernou fyziku, budou nás zajímat ještě další charakteristiky částice – její hmotnost, náboj, stupeň excitace, atd. Tím by ale datový soubor narůstal na neúměrnou velikost (mějme na paměti, že v některých programech se pracuje s 10^6 a více částicemi), proto je vhodné data optimalizovat tak, že se zavede jen jeden další integrální parametr „typ částice“ a přiřazení konkrétních hmotností, atd. se provede v programu jen jednou.

Pracovní oblast

Při částicovém modelování, jak při použití metody Monte Carlo tak i metody molekulární dynamiky, musíme nejprve určit oblast, v níž budeme studovaný proces simulovat. Při modelování transportu metodou Monte Carlo budeme mít tuto oblast tvořenou třemi částmi: zdrojem částic, vlastní pracovní oblastí v níž se transport odehrává a cílovou oblastí – viz obr. 4.8.



Obrázek 4.8: Znázornění pracovní oblasti při modelování transportu částic.

Vlastní program simulující průchod částic hmotným prostředím pak vypadá tak, že ze zdroje vstoupí do pracovní oblasti jedna částice – výklad

budeme pro jednoduchost provádět pro jednočásticovou metodu. Pohyb částice v pracovní oblasti budeme detailně sledovat. Nejprve určíme vzdálenost do bodu interakce s materiálem, pak určíme typ interakce a podle něj upravíme směr a rychlost dalšího pohybu částice nebo částic (dojde-li totiž při některé interakci k vytvoření dalších částic, i původně jednočásticová metoda může přejít v metodu mnohočásticovou). Pak celý postup opakujeme a současně testujeme, zda trajektorie nějakým způsobem nebyla ukončena.

Zdroj částic představuje generátor částic, které vstupují do naší pracovní oblasti. Jeho charakteristiku budeme přebírat ze studovaného fyzikálního jevu – může to být např. úzký svazek monoenergetických částic dopadajících kolmo na hranici pracovní oblasti při studiu procesů v elektronovém mikroskopu, krychlová oblast s náhodně rozmístěnými elektrony a ionty pohybující se s Maxwellovým rozdělením rychlostí při modelování v plazmatu, bodový zdroj se sférickou symetrií, apod. Z hlediska datové struktury znázorněné na obr. 4.7 provádí zdroj (tj. ta část programu, v níž je zdroj realizován) naplnění příslušných sloupců dat pro částice vstupující ze zdroje počátečními hodnotami, tj. hodnota proměnné x se nastaví na nulu, zadají se případné hodnoty dalších prostorových souřadnic y a z, a zadají se počáteční rychlosti nebo energie těchto částic.

Pracovní oblast modelu je ta část prostoru, kde dochází k interakcím procházejících částic s látkovým prostředím. Tyto interakce budou vždy v modelu přítomné, neboť pokud by tam nebyly, mohli bychom problém řešit analyticky a nepotřebovali bychom používat techniky počítačového modelování. Charakteristiku prostředí, v němž transport probíhá, musíme převzít z vnějšku modelu, neboť to jsou základní experimentální údaje, jimiž obecné transportní schéma přizpůsobíme konkrétní studované problematice. Z literatury nebo od kolegů-experimentátorů musíme získat informace o počtu rozptylových procesů, jejich intenzitě a přesný popis jejich vlivu na procházející částice. Tyto údaje shrneme do jedné makroskopické veličiny – výsledné střední volné dráhy (nebo celkového účinného průřezu) – charakterizující kvantitativně celkový průměrný vliv prostředí na částici. Na jejím základě vytvoříme tzv. náhodnou volnou dráhu, po jejímž uražení se částice srazí a trajektorie znázorněná na obr. 4.8 se zalomí. Pak musíme z existujících rozptylových procesů vybrat typ konkrétní interakce, podle její charakteristiky změnit rychlost částice $\vec{v}$, postup zopakovat a takto vytvořit celou trajektorii. Detailní popis jednotlivých kroků bude uveden v dalších sekcích. Z hlediska datové struktury se při pohybu v pracovní oblasti budou měnit všechny údaje charakterizující částici – po uražení náhodné volné dráhy změníme o uražený úsek trajektorie prostorové souřadnice a po rozhodnutí o typu interakce upravíme odpovídajícím způsobem údaje o rychlosti nebo energii částice. Pokud dojde k tzv. štěpení trajektorie, tj. pokud se při určité srážce vytvoří další

částice (fyzikálně to odpovídá např. štěpení atomů v jaderné fyzice nebo ionizaci atomů ve fyzice plazmatu), zavedeme do datového souboru další částici nebo částice.

Trajektorie může skončit třemi způsoby – standardním vstupem do *cílové oblasti*, ukončením trajektorie záchytem a případně i návratem částice do zdrojové oblasti. Ve všech třech případech zpracování této informace závisí opět na studovaném jevu a na vytvořeném modelu. Obvyklé je, že pouze zapisujeme počet částic, které pronikly do cílové oblasti a srovnáváme je s počtem částic původně zdrojem vypuštěných (pak druhé dva způsoby ukončení trajektorie znamenají dva možné ztrátové procesy, které podle potřeby dále analyzujeme). Je ovšem možné provádět i detailní analýzu částic vyletujících z pracovní oblasti a zaznamenávat jejich rychlosti $\vec{v_i}$, což pak využijeme např. při modelování úhlového nebo energetického rozdělení studovaných částic.

Závěrem této sekce ještě jednu poznámku: V modelu transportního jevu mluvíme o třech oblastech – zdrojové, pracovní a cílové, které důsledně oddělujeme jak v popisu modelu tak i v programu (formou samostatných programových jednotek, podprogramů, bloků nebo modulů – viz sekce 2.3 „Zásady strukturovaného programování“). To někdy odpovídá i fyzikální realitě ve studovaném jevu – např. při modelování procesů v atomovém reaktoru je zdrojovou oblastí jeden palivový článek, pracovní oblastí moderátor zpomalující procházející neutrony a cílovou oblastí další palivový článek. Jinde však nelze tyto tři oblasti prostorově odlišit – např. při modelování procesů v nízkoteplotním plazmatu, přesto však při vytváření modelu a jeho realizaci na počítači budeme tyto tři oblasti oddělovat. Z tohoto hlediska se jedná ne o prostorové, ale o časové oddělení – částice jsou nejprve generovány, pak dochází k transportu a nakonec se výsledky zpracují.

Rozptylové procesy

Předpokládejme, že v látce, v níž probíhá transport částic, je přítomno celkem k typů rozptylových procesů. Tyto procesy jsou charakterizovány středními volnými drahami $\lambda_1, \lambda_2, \ldots, \lambda_k$, případně účinnými průřezy jednotlivých typů interakcí $S_1, S_2, \ldots, S_k$. Střední volná dráha λ_i je průměrná vzdálenost mezi srážkami i-tého typu a udává se v metrech. S účinným průřezem je to trochu složitější. Můžeme jej zavést dvěma způsoby, jako makroskopický účinný průřez (vztažený k počtu záchytných center v jednotce objemu)

$$S_i = \frac{1}{N \cdot \lambda_i},$$

kde $N = 3,21\,10^{22}$ molekul nebo atomů v m^3 při $T = 300\,K$. Tento účinný průřez je udáván v m^2. Druhou možností je mikroskopický účinný průřez

$$S_i = \frac{1}{\lambda_i}$$

udávaný v jednotkách m^{-1}. Pokud dostaneme experimentální hodnoty, tak rozměr účinného průřezu a jeho číselná velikost nám řekne, která definice byla při měření použita. My v dalším výkladu budeme pro jednoduchost pracovat s účinným průřezem udávaným v m^{-1}.

Než budeme pokračovat v modelování, z dílčích interakcí vytvoříme celkový účinný průřez S nebo celkovou střední volnou dráhu λ. Složení dílčích interakcí provedeme podle vztahů

$$S = \sum_{i=1}^{k} S_i\,,$$

nebo

$$\frac{1}{\lambda} = \sum_{i=1}^{k} \frac{1}{\lambda_i}\,.$$

Dílčí i celkové střední volné dráhy mohou být konstantami, většinou však závisejí na poloze (v nehomogenním prostředí) nebo na energii částice. Totéž platí pro účinné průřezy.

Konkrétní typy rozptylových procesů, jejich intenzity vyjádřené pomocí středních volných drah λ_i a účinných průřezů S_i, a energetické a prostorové závislosti těchto veličin jsou parametry modelování, kterými přizpůsobíme obecný model studovanému jevu. Základní typy rozptylových procesů jsou následující:

- Pružný rozptyl
  Při pružném rozptylu se při interakci nemění celková energie interagujících částic. Ve fyzice pevných látek, kde mříž má prakticky nekonečnou hmotnost proti interagující částici, to znamená, že energie částice E před i po srážce bude stejná. Směr pohybu částice se ovšem změní a proto se změní i „složky“ energie E_x, E_y a E_z. Naproti tomu např. ve fyzice plazmatu, i při srážce lehkého elektronu s těžkým iontem se při přesném výpočtu musí určitá malá ztráta energie elektronu (úměrná podílu hmotností obou částic) započítat. Směr částice po interakci je buď zadán simulovanými experimentálními podmínkami, častěji se však předpokládá úhlově izotropní rozptyl, kde směrové kosíny se rozehrávají ze vztahů (4.30).

- Nepružný rozptyl
  Při nepružném rozptylu vždy dochází ke ztrátě energie ΔE. Tato ztráta energie může být konstantní (např. při excitaci atomu do vyššího stavu je ΔE rovno rozdílu energetických hladin atomu), může to být ale i náhodná veličina (při ionizaci, kdy ΔE je rovno ionizační energii zvýšené o hodnotu energie, kterou si odnese uvolněný elektron). Směr částice po interakci bývá většinou zadán z experimentu, někdy ale při nedostatku experimentálních dat se opět uchylujeme ke vztahům (4.30). V řadě modelů bývá i větší počet nepružných rozptylových procesů s různými parametry.

- Záchyt
  Záchyt fyzikálně odpovídá různým mechanismům, např. pohlcení neutronu, záchyt elektronu na pasti v zakázaném pásu dielektrika, atd. V modelu se to ošetřuje stejně – trajektorie prostě v daném bodě končí a ze zdroje je vypuštěna další částice. Pokud to ale studovaný jev vyžaduje, lze model doplnit i o popis fyzikálních procesů vyvolaných záchytem částice a uvolněním energie nebo náboje, kterou nese – např. po zabrzdění rychlého elektronu v látce vzniká elektromagnetické záření nebo se zvýší koncentrace náboje v daném místě materiálu.

- Štěpení
  Štěpení znamená proces, při kterém z jedné primární studované částice vznikají dvě nebo i více sekundárních – neutronů v jaderné fyzice, elektronů v plazmatu, apod. Programovací problém se štěpením spojený je, že pokud je tento proces intenzivní, často se opakuje a z původní jedné částice vzniká celá lavina. To sice odpovídá fyzikální realitě (řetězová reakce, elektrický průraz, jiskra nebo blesk), velmi obtížně se to však programuje. Proto tomuto rozptylovému procesu věnujeme samostatný paragraf.

Po uražení náhodné volné dráhy dojde určitě k interakci, je však třeba rozhodnout, která interakce to bude. Při tom se využijí hodnoty středních volných drah dílčích interakcí λ_i (nebo účinných průřezů S_i) k nalezení pravděpodobností dílčích interakcí v daném místě a při dané energii částice. Pro pravděpodobnost výskytu i-té interakce použijeme jeden ze vztahů

$$p_i = \frac{\lambda}{\lambda_i}$$
$$p_i = \frac{S_i}{S}.$$

Pokud tyto pravděpodobnosti známe, použijeme standardní postup na rozehrávání diskrétní náhodné veličiny (zobrazený na obr. 4.2).

Náhodná volná dráha

Základním úkolem při řešení transportního problému ve fyzice je rozehrání náhodné volné dráhy. V této souvislosti musíme operovat se dvěma pojmy: *náhodná volná dráha* ξ je vzdálenost, kterou urazí částice mezi dvěma po sobě jdoucími interakcemi, a jedná se o náhodnou veličinu. Naproti tomu *střední volná dráha* λ je obvyklé (nenáhodné) číslo udávající průměrnou vzdálenost mezi interakcemi a charakterizující tak prostředí, v němž transport probíhá. Střední volná dráha představuje první moment náhodné veličiny ξ a je mezi nimi proto obvyklý vztah (4.16)

$$\lambda = E\xi \doteq \frac{1}{n} \cdot \sum_{i=1}^{n} \xi_i \,.$$

Nyní však před námi stojí složitější problém. Zákon velkých čísel (4.16) nám umožňuje přibližně řešit tuto implikaci pro obecnou náhodnou veličinu ξ:

$$\xi_1, \xi_1, \ldots, \xi_n \quad \Rightarrow \quad E\xi \,,$$

naproti tomu opačná implikace, kterou potřebujeme vyřešit pro generování náhodných volných drah:

$$\lambda = E\xi \quad \Rightarrow \quad \xi_1, \xi_1, \ldots, \xi_n$$

není na obecné úrovni řešitelná.

Pokud využijeme fyzikální vlastnosti rozptylu částic v látce, lze vztah pro generování náhodných volných drah odvodit. Musí však být splněn předpoklad, že střední volná dráha je konstantní, tj. látkové prostředí je homogenní a střední volná dráha nezávisí ani na dalších parametrech částice (především energii). Za tohoto dosti silného předpokladu, můžeme použít pro generování jednotlivých realizací náhodných volných drah ξ_i jednoduchý vztah

$$\xi_i = -\lambda \cdot \ln \gamma_i \,, \tag{4.45}$$

kde γ je rovnoměrně rozdělená náhodná veličina v intervalu $(0, 1\rangle$ – hodnota 0 nesmí být mezi generovanými hodnotami zastoupena. Pokud ale není předpoklad $\lambda = konst.$ splněn, vztah (4.45) nemůžeme použít. Lze sice odvodit silnější vztah platný i pro nekonstantní střední volné dráhy, jeho použití v metodě Monte Carlo je však téměř vyloučené, neboť je velmi neefektivní. Místo toho je třeba použít umělý postup popsaný v sekci 4.5.2.

Vztah pro rozehrání náhodné volné dráhy ξ můžeme odvodit tímto postupem: Předpokládejme, že částice se pohybuje z bodu $x = 0$ podél osy x a po cestě se může srážet. Zavedeme parametr S – celkový

účinný průřez interakce částice s prostředím. Rozdělovací funkce náhodné volné dráhy ξ, $F(x)$, bude

$$F(x) = P\{\xi < x\}\,.$$

Pravděpodobnost, že částice bude mít první srážku v intervalu $\langle x, x+\Delta x\rangle$, můžeme určit dvěma způsoby. Za prvé, pro Δx dosti malé s využitím distribuční funkce dostaneme

$$F(x+\Delta x) - F(x)\,.$$

Za druhé, můžeme tutéž pravděpodobnost vyjádřit pomocí účinného průřezu S

$$[1 - F(x)] \cdot S \cdot \Delta x\,,$$

kde $1-F(x)$ je pravděpodobnost, že částice doletí bez srážky do místa x a $S \cdot \Delta x$ je pravděpodobnost srážky v intervalu $\langle x, x+\Delta x\rangle$.

Vytvoříme-li z obou těchto vztahů pro pravděpodobnost srážky rovnici a provedeme-li limitní přechod $\Delta x \to 0$, dostaneme pro rozdělovací funkci $F(x)$ výraz

$$F(x) = 1 - \exp\left[-\int_0^x S(s)\,\mathrm{d}s\right].$$

Odpovídající hustota pravděpodobnosti rozdělení náhodných volných drah ξ bude

$$p(x) = S(x) \cdot \exp\left[-\int_0^x S(s)\,\mathrm{d}s\right]. \tag{4.46}$$

Náhodná volná dráha je tedy spojitou náhodnou veličinou s oborem hodnot $x \in \langle 0, \infty)$ a hustotou pravděpodobnosti (4.46). Při znalosti celkového účinného průřezu $S(x)$ můžeme tyto údaje využít k rozehrávání konkrétních hodnot náhodných volných drah. Problém je ale ve tvaru závislosti (4.46) – jedná se o integrál v exponenciále. Pokud bude možno tento integrál spočítat analyticky, lze pro rozehrávání hodnot náhodné volné dráhy $\xi_1, \xi_2, \ldots$ připravit explicitní vzorec typu (4.45) a zařadit jej do programu. Jediný problém bude, že to nelze udělat obecně, ale jen při znalosti konkrétních rozptylových procesů v našem modelu. Pokud ovšem nebude integrál ve vztahu (4.46) řešitelný analyticky, v modelování metodou Monte Carlo jej nemůžeme použít – případné numerické řešení by bylo příliš pomalé, neboť v jednom programu bychom je museli provádět 10^6 až 10^9-krát, a někdy i vícekrát.

V prostředí, kde celkový účinný průřez S bude konstantní (včetně závislosti na parametrech částice), dostaneme zjednodušené výrazy pro rozdělovací funkci a hustotu pravděpodobnosti

$$\begin{aligned} F(x) &= 1 - \exp(-S\,x) \\ p(x) &= S \cdot \exp(-S\,x)\,. \end{aligned}$$

Metodou inverzní funkce (4.26) získáme pro rozehrání náhodné volné dráhy ξ vztah (4.45)

$$\xi_i = \frac{1}{S} \cdot \ln \gamma_i = -\lambda \cdot \ln \gamma_i \,.$$

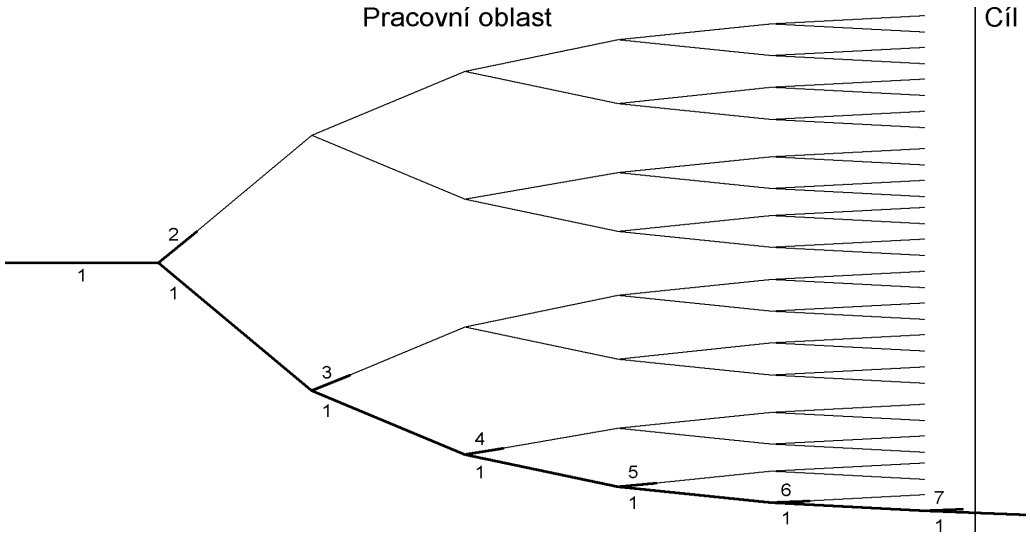
Štěpení trajektorie

V modelech některých fyzikálních jevů se mezi rozptylovými procesy vyskytují takové, při kterých vznikají nové částice (štěpení atomů v jaderné fyzice, ionizace atomů a molekul ve fyzice plazmatu, atd.) a tak se trajektorie štěpí a studovaný model začne být mnohočásticový.

Při štěpení trajektorie se z původní jedné stopy částice vytvoří tzv. „strom“ a my musíme v programu projít všemi jeho větvemi. Nejjednodušší a nejméně efektivní metoda by postupně simulovala všechny interakce vytvářející strom, všechny vznikající větve by uložila do paměti počítače a pak by je postupně analyzovala. K zefektivnění celé procedury byly navrženy dvě metodiky procházení stromovou strukturou:

– Analýza po větvích

– Analýza po generacích



Obrázek 4.9: Strom trajektorií částice – zpracování po větvích.

Princip *analýzy po větvích* je znázorněn na obr. 4.9. V tomto obrázku jsme pro jednoduchost předpokládali, že se každá trajektorie při štěpení dělí

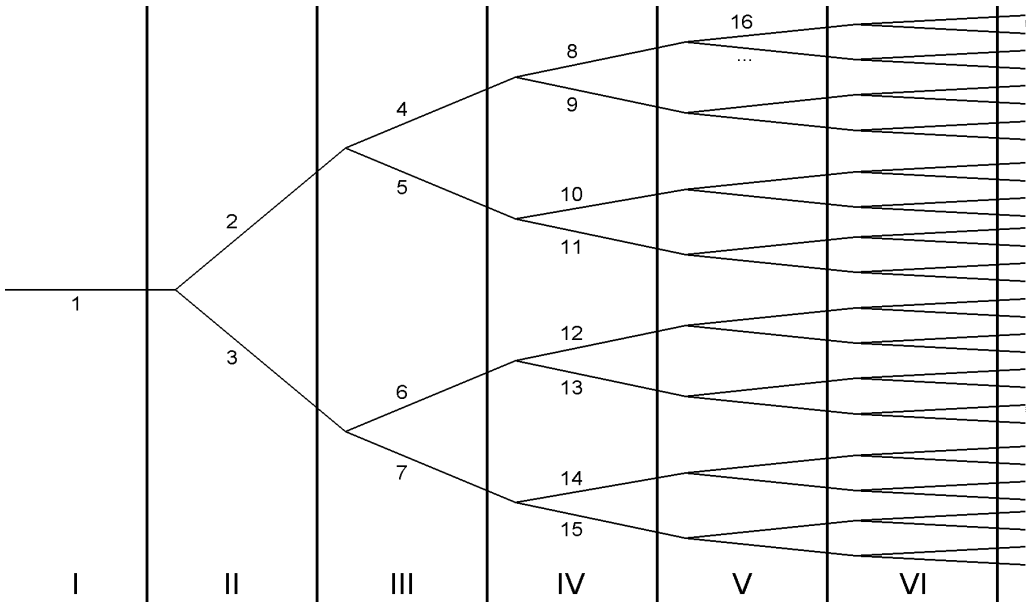
na dvě, že jsou trajektorie mezi štěpeními stejně dlouhé a z obrázku jsme vynechali další srážky nevedoucí k štěpení trajektorie.

Na počátku budeme mít pouze částici číslo 1. Při první srážce vznikne další částice. My i nadále budeme jednu (původní) částici nazývat číslem 1 a druhou částici číslem 2. Informaci o vzniku částice 2 uložíme do paměti počítače – do datové struktury (4.7) (tam uložíme počáteční data nové částice, tj. místo jejího vzniku a její rychlost nebo energii, případně i další parametry) – a dále pokračujeme ve sledování částice 1. Ta při další interakci vytvoří částici 3, tu opět uložíme do datové struktury a pokračujeme ve sledování původní částice. Tak postupně vzniknou částice 4, 5, 6 a 7. Když částice číslo 1 vyletí z pracovní oblasti, zpracujeme ji, tj. uložíme informaci např. o energii nebo úhlu vyletující částice. V tomto okamžiku je ze stromové struktury známa pouze část zakreslená na obr. 4.9 zvýrazněně. Po zpracování částice číslo 1 zrušíme část stromu, tj. větev vycházející z uzlu 1×7 do cílové oblasti, a současně vymažeme i data částice 1 z datové struktury. Jelikož při převážně většině interakcí nedokážeme od sebe původní a nově vytvořené částice odlišit, budeme se řídit jednoduchým pravidlem, že původní částice se vždy odklání jedním směrem - na obr. 4.9 např. vpravo, tj. dolů.

Pak budeme pokračovat ve vytváření a zpracování stromu. Z datové struktury vezmeme poslední uloženou částici, tj. částici číslo 7 a budeme pokračovat v započaté větvi. Když tato částice vyletí z pracovní oblasti, zaznamenáme její parametry a opět smažeme část stromu až k předchozímu uzlu (tj. uzlu 1×6) včetně údajů o částici 7 v datové struktuře. Opět budeme pokračovat ve větvi vycházející z tohoto uzlu. Pokud se trajektorie bude znovu větvit, uložíme jednu částici do datové struktury a budeme pokračovat ve sledování částice číslo 6.

Pokud budeme tento postup důsledně uplatňovat, postupně projdeme celou stromovou strukturu. Abychom optimalizovali práci s pamětí počítače, je třeba pravidelně odstraňovat prázdné sloupce z datové struktury a částice přečíslovávat.

Alternativní princip *analýzy po generacích* je znázorněn na obr. 4.10. I tento obrázek je kreslen zjednodušeně podobně jako obr. 4.9. V tomto algoritmu se vstupující částice číslo 1 nezachovává, ale po první srážce se mění na dvě částice s čísly 2 a 3 – tomu opět odpovídá vymazání údaje o částici 1 z datové struktury a zavedení dvou nových sloupců ve struktuře. Pak postupně sledujeme obě vzniklé větve a vytvoříme nové částice – ze dvou částic 2 a 3 vzniknou celkem čtyři nové částice s čísly 4, 5, 6 a 7, opět s odpovídajícími změnami údajů v datové struktuře. Po dalších srážkách vznikne osm částic, atd. Tyto postupně vznikající částice budeme řadit do tzv. generací, tj. máme jednu částici generace I, dvě částice generace II, čtyři částice generace III, osm částic generace IV, atd. Důležité na popisovaném algoritmu



Obrázek 4.10: Strom trajektorií částice – zpracování po generacích.

je, že dříve než přejdeme k částicím další generace, postupně zpracujeme všechny větve generace předchozí. V datové struktuře je třeba zaniklé částice předchozích generací pravidelně vymazávat a uchovávat v této struktuře pouze částice stejné generace. Pokud navíc budeme po zpracování všech částic příslušné generace datovou strukturu upravovat tak, abychom odstranili všechny prázdné sloupce, bude mít datová struktura rozumnou velikost. Rozměr datové struktury bude roven počtu částic poslední generace před jejich opuštěním pracovní oblasti.

Algoritmus analýzy po větvích se přednostně používá v prostředí, kde je relativně málo srážek vedoucích ke štěpení trajektorie, tj. v případě, kdy vytvořený strom je řídký a s dlouhými větvemi. Naproti tomu algoritmus analýzy po generacích se používá zejména v případě stromů s velkým počtem krátkých větví, které popisují vznik lavin částic.

> I když umíme modely se štěpením trajektorie vyřešit, přináší jejich programování řadu obtíží, takže je výhodnější se takovým modelům vyhnout. V některých případech to udělat nemůžeme – pokud je např. cílem modelu studovat vznik laviny nabitých částic ve tvaru jiskry nebo blesku, musí samozřejmě model i program tuto lavinu obsahovat. Je však řada problémů, kde ke štěpení trajektorie dochází, ale není to těžiště studovaného jevu. Příkladem mohou být veškeré procesy v nízkoteplotním plazmatu. Bez procesu ionizace, což je fyzikální ekvivalent štěpení trajektorie, plazma nemůže vzniknout a již

existující plazma by v krátké době zaniklo rekombinací nabitých částic. Velká řada problémů se ale studuje v rovnovážném stavu, kdy je nárůst částic vlivem štěpení kompenzován nějakým jiným procesem – většinou tzv. ambipolární difuzí nabitých částic ke stěnám s následnou rekombinací na stěně. Pokud tedy studujeme jiné jevy, které v plazmatu probíhají (např. chemické reakce, apod.), lze z modelu vypustit proces ionizace spolu s kompenzujícím procesem ambipolární difuze a výsledek se nezmění, přičemž námaha při sestavování programu se zmenší. Tento postup je blízký umělým obratům v metodě Monte Carlo popsaným v další sekci.

4.5.2 Modelování fyzikálních procesů se zvýšenou účinností

Jak jsme již dříve uvedli, při počítačovém experimentu můžeme postupovat dvěma cestami – pomocí přirozených nebo umělých modelů. Vše, co jsme zatím uvedli, se týkalo přirozených modelů, v kterých jsme se snažili popsat studovaný jev pomocí modelu co nejpřesněji, aby se i výsledky modelování co nejméně lišily od výsledků reálných procesů. Tento oprávněný požadavek ale často vede na modely příliš neefektivní, takže je nelze s dostupnou výpočetní technikou v dostupném čase vyřešit. Proto byla navržena metodika modelování pomocí umělých obratů. Pro chybu metody Monte Carlo jsme si odvodili vztah (4.31) $0,67 \cdot \sqrt{D\xi/n}$. Podle něj lze rychlost simulování metodou Monte Carlo zvýšit tak, že zvolíme pro popis studovaného jevu jinou náhodnou veličinu ξ s menším rozptylem $D\xi$.

Umělý obrat je takový předpoklad, který vede k vytvoření náhodného procesu s menším rozptylem, i když tento proces nepopisuje dostatečně dobře studovaný jev. Stručně tedy lze umělý obrat charakterizovat tak, že s jeho zavedením vyměníme přesnost simulace za její rychlost. Abychom ale přesnost simulace zachovali, součástí umělého obratu musí být i návod, jak ze získaného nesprávného výsledku dodatečně určit výsledek správný.

Zavedení statistických vah

Základním obratem, zrychlujícím konvergenci studovaného procesu, je zavedení váhových faktorů. Tyto *statistické váhy* představují další parametr částice. Nemají sice přímý fyzikální význam, ale s jejich pomocí lze ze stejně rozsáhlého počítačového experimentu získat více informací.

K tomuto účelu poněkud zmodifikujeme datovou strukturu zavedenou na obr. 4.7. Vedle údajů o poloze a rychlosti (či energii) částice budeme ještě zaznamenávat okamžité hodnoty statistické váhy částice w - viz obr. 4.11.

	1	2	3	4	5	...												
x																		
y																		
z																		
v_x																		
v_y																		
v_z																		
typ																		
w																		

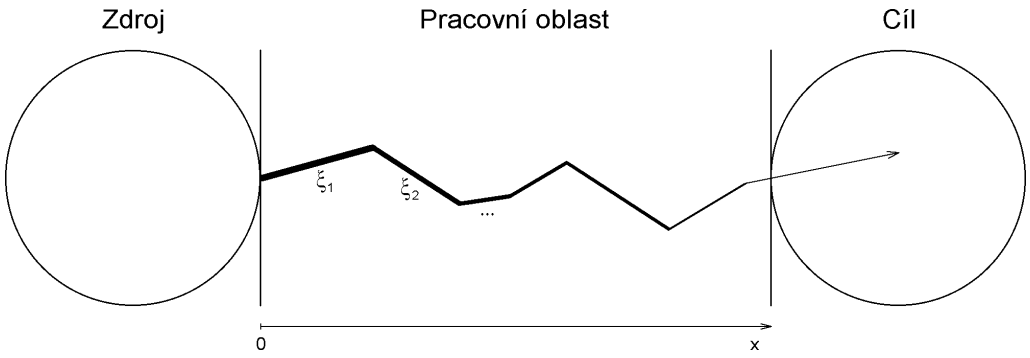
Obrázek 4.11: Datová struktura pro částicové modelování – umělé modely pracující se statistickou váhou částic.

První použití umělého obratu bylo v jaderné fyzice, kde byl modelován transport částic silně absorbujícím prostředím – např. průchod produktů štěpné reakce z aktivní zóny atomového reaktoru stěnou, jež tyto částice pohlcuje. Bude-li úkolem počítačového experimentu navrhnout tvar a složení této stěny tak, aby utlumila tok procházejících částic na bezpečnou míru, cílem modelování bude vypočítat koeficient záchytu této stěny. Je zřejmé, že prostředí tvořící stěnu musí obsahovat vedle jiných srážkových procesů i záchyt částic.

Nejprve si popišme modelování pomocí přirozených procesů. Bude se jednat o klasické transportní schéma znázorněné na obr. 4.8 – zdrojem bude aktivní zóna reaktoru, pracovní oblastí ochranná stěna jejíž tloušťku a složení máme navrhnout, a cílovou oblastí bude okolí atomového reaktoru. V pracovní oblasti budeme předpokládat celkem k typů srážkových procesů popsaných středními volnými drahami $\lambda_1, \lambda_2, \ldots, \lambda_k$. Mezi těmito srážkovými procesy bude i záchyt částice – nechť to je poslední, k-tý, proces. Použijeme nejjednodušší jednočásticové transportní schéma: Do pracovní oblasti ze zdroje postupně vypustíme N_1 částic, budeme rozehrávat jejich náhodné volné dráhy, testovat typ srážky a podle něj upravovat parametry částice. Trajektorie může končit dvěma způsoby – buď částice stěnou projde nebo někde v průběhu simulace dojde k záchytu a pak trajektorii ukončíme. Pokud částice projde až do cílové oblasti, hodnotu počitadla N_2 zvýšíme o jednotku. Hledaný koeficient záchytu K_z budeme odhadovat pomocí jednoduchého vztahu

$$K_z = \frac{N_2}{N_1}\,.$$

Abychom získali výsledek s dostatečnou přesností, je třeba v modelování pokračovat tak dlouho, pokud stěnou neprojde aspoň $N_2 \simeq 1 \cdot 10^3$ částic. Pokud by se jednalo o obvyklý materiál, kde záchyt je jeden z řady rovnocenných procesů, byla by to výpočetně jednoduchá úloha. Při dané formulaci problému však tomu bude jinak. Pokud má být stínění reaktoru dostatečně účinné, musí být koeficient záchytu K_z velmi malý (nejméně $1 \cdot 10^{-6}$, spíše však ještě mnohem menší). To však znamená, že do stěny musí vstupovat nejméně $N_1 = 1 \cdot 10^9$ částic a každá z nich před zachycením v materiálu stěny prochází řadou dalších interakcí. Tím se úloha stává výpočetně velmi složitou – v padesátých letech, kdy byl tento problém poprvé studován, to bylo zcela neřešitelné (a i současná výpočetní technika by to zvládla jen s obtížemi).



Obrázek 4.12: Transport částic absorbujícím prostředím s využitím statistické váhy částice simulující záchyt.

Nyní se popišme stejný proces modelovaný s využitím umělého obratu, a to v původní historické formulaci. Prostředí, které mělo původně k typů srážkových procesů, bude nyní obsahovat pouze $k-1$ typů srážek, tj. záchyt z modelu vynecháme. Budeme předpokládat, že ze zdroje do pracovní oblasti najednou vstupuje místo jedné částice celý „balík“ w_0 částic. Rozehrajeme náhodnou volnou dráhu, která bude společná pro všechny částice v balíku. Po jejím uražení se všechny částice srazí. Při rozehrávání typu srážky budeme volit pouze mezi $k-1$ možnostmi, mezi nimiž nebude záchyt, takže trajektorie balíku bude určitě pokračovat. Předpokladem modelu je, že se při srážkách balík nerozpadne, takže částice se budou pohybovat i nadále spolu. Postup budeme opakovat tak dlouho, pokud balík (nebo to, co z něj zbyde) nevyletí ven. Pokud bychom ale problém formulovali pouze takto, byl by výsledek nevratně špatný a proto nyní musíme záchyt do modelu dodatečně zahrnout. Budeme předpokládat, že po uražení náhodné volné dráhy ξ_1 se v místě první srážky balík částic zmenší a dále bude pokračovat pouze w_1 částic, kde

$w_1 = w_0 \cdot \exp(-\xi_1/\lambda_k)$ a λ_k je střední volná dráha pro záchyt. Po uražení další náhodné volné dráhy ξ_2 při další srážce se balík částic opět zmenší atd., takže na konci trajektorie vystoupí do cílové oblasti pouze – viz obr. 4.12

$$w_{cil} = w_0 \cdot \exp\left[-(\xi_1 + \xi_2 + \cdots + \xi_m)/\lambda_k\right]$$

částic. Abychom získali správný výsledek, musíme nově definovat i koeficient záchytu jako

$$K_z = \frac{1}{N\, w_0} \cdot \sum_{i=1}^{N} w_i \,,$$

kde w_i je váha i-té částice při vstupu do cílové oblasti a N je počet částic vyslaných zdrojem.

Je zřejmé, že tento modifikovaný algoritmus je nesrovnatelně efektivnější než přirozená metoda, neboť každá vstupující částice (resp. balík částic) projde celou pracovní oblastí a k dosažení přesnosti stejné jako v předchozím případě stačí proto pracovat pouze s $N = 1 \cdot 10^3$ částicemi. Manipulace s počty částic v balíku sice poněkud výpočet zpomalí, ale úspora v počtu částic nesrovnatelně převažuje. V současné době je považována formulace pomocí balíku částic za poněkud nepřirozenou (a též, pokud jsme na počátku nevhodně zvolili velikost balíku w_0, může trajektorie v pracovní oblasti přece jenom skončit, a to vyčerpáním všech částic). Proto se místo o balíku částic mluví o jediné částici, které se přiřadí další vlastnost a to statistická váha. Tato váha je na počátku rovna jedné ($w_0 = 1,0$) a v průběhu transportu klesá. Koeficient záchytu je pak určován jako průměrná zbytková váha částice po průchodu pracovní oblastí.

> Z hlediska terminologie, kterou teprve budeme probírat, tento postup představuje tzv. *hybridní* modelování, kdy problém řešíme současně dvěma technikami – z původních k typů interakci modelujeme prvních $k-1$ typů stochasticky metodou Monte Carlo a poslední interakci modelujeme spojitou technikou.

Metoda nulové srážky

Z velkého množství dalších známých umělých obratů má v počítačové fyzice při studiu transportu částic mimořádné místo tzv. metoda nulové srážky. Tato metoda byla navržena v roce 1972 [12] a s její pomocí můžeme i v nehomogenních prostředích používat pro rozehrávání náhodné volné dráhy jednoduchý vzorec (4.45)

$$\xi_i = -\lambda \cdot \ln \gamma_i$$

místo podstatně složitějšího a výrazně pomalejšího postupu (4.46).

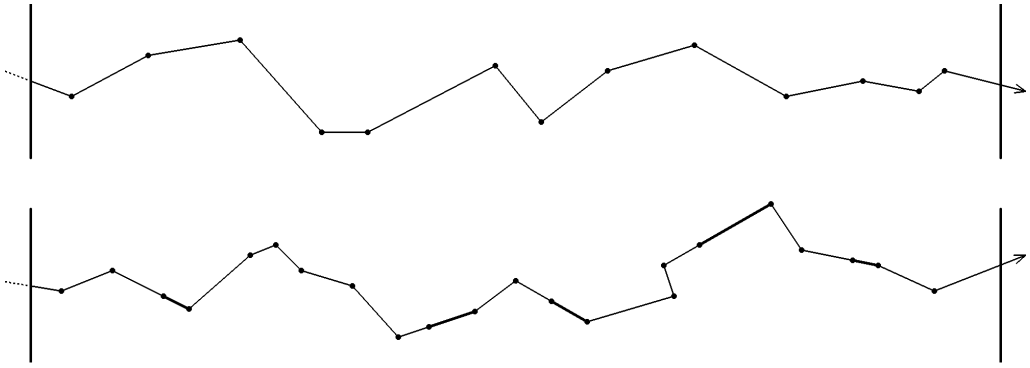
Opět jako v předchozím případě budeme měnit počet interakcí v pracovní oblasti. Původně mělo studované prostředí k typů srážkových procesů charakterizovaných účinnými průřezy $S_1, \ldots, S_k$. Z nich byl vytvořen celkový účinný průřez S a celková střední volná dráha λ. Tyto veličiny ale závisely na poloze částic – ať již vlivem nehomogennosti prostředí a nebo vlivem změny energie částice – a proto nebylo možno využít pro rozehrávání náhodné volné dráhy vztah (4.45). Proto v nové formulaci modelu budeme pracovat s prostředím obsahujícím $k+1$ typů srážkových procesů charakterizovaných účinnými průřezy $S_1, \ldots, S_k, S_{k+1}$. Ten nový $k+1$. srážkový proces jsme si vymyslili, proto si můžeme navrhnout i jeho závislost na poloze, na energii nebo na jiném fyzikálním parametru způsobujícím nekonstantnost původního účinného průřezu. Uděláme to tak, aby nový celkový účinný průřez

$$S' = S_1 + \cdots + S_k + S_{k+1}$$

byl konstantní. Pak i upravená střední volná dráha λ' bude konstantní a pro rozehrávání náhodné volné dráhy ξ'_i můžeme používat jednoduchý vztah

$$\xi'_i = -\lambda' \cdot \ln \gamma_i \,.$$

Každému srážkovému procesu přísluší i určitá fyzikální charakteristika, podle níž měníme energii a směr částice po srážce. Tomuto novému umělému procesu, který jsme do modelu zavedli, se říká *nulová srážka*, protože nebude měnit ani energii interagující částice ani směr jejího pohybu – další část trajektorie bude prodloužením předchozího úseku. Tento předpoklad plně postačí k tomu, aby kompenzoval vliv zavedeného umělého obratu, což si můžeme demonstrovat na obr. 4.13.



Obrázek 4.13: Modelování transportního procesu částic pracovní oblastí: nahoře – přirozený algoritmus, dole – simulace s užitím metody nulové srážky.

V případě přirozeného procesu je trajektorie tvořena náhodnými volnými drahami ξ_i rozehrávanými na základě střední volné dráhy λ. V případě umělého procesu pracujeme s náhodnými volnými drahami ξ_i' rozehrávanými pomocí střední volné dráhy λ'. Jelikož čárkované veličiny odpovídají mechanismu s přídavnou interakcí, bude $\lambda' < \lambda$ a proto i náhodné volné dráhy ξ_i' budou statisticky kratší než ξ_i a čárkovaná trajektorie bude tvořena větším počtem kratších úseků. Mezi tyto úseky však budou náhodně vkládány úseky odpovídající mechanismu nulové srážky, které budou opět náhodně prodlužovat ostatní úseky. Proto při vyhodnocení původní (obr. 4.13 nahoře) i čárkované (obr. 4.13 dole) trajektorie budou obě trajektorie statisticky stejné a z metody nulové srážky se projeví jen výrazná úspora času modelování.

Další umělé obraty

V různých oblastech počítačové fyziky bylo navrženo velké množství umělých obratů a další stále ještě vznikají. Mezi ně patří např.

- Váhy nahrazující výlet z pracovní oblasti
  V určitém bodě trajektorie rozehrajeme nový směr a určíme vzdálenost v tomto směru ke kraji oblasti t. Při rozehrání nové náhodné dráhy je určitá pravděpodobnost toho, že částice již pracovní oblast opustí. Pokud tomu chceme z nějakého důvodu zabránit, rozehrajeme novou náhodnou dráhu pomocí upraveného vztahu (4.45)

  $$\xi_i = -\lambda \cdot \ln\left[1 - \gamma_i \cdot (1 - e^{-S\,t})\right].$$

  Současně však musíme zmenšit statistickou váhu částice o tu část původního „balíku“, která pracovní oblast opustila

  $$w' = w \cdot (1 - e^{-S\,t}).$$

  Tento postup se však používá méně často, neboť časové ztráty vznikající úpravou statistické váhy a rozehráváním omezené náhodné volné dráhy zmenšují výhody získané rychlejší konvergencí metody.

- Rozdělení trajektorie
  Na základě statistických vah můžeme podle potřeby v kterékoliv části pracovní oblasti zpřesnit výpočet. K tomu postačí v uvedeném prostoru zavést umělou interakci typu štěpení trajektorie a předpokládat, že po srážce jedné částice o statistické váze w_i vznikne m nových částic o vahách w_i/m a dále jejich trajektorie studovat odděleně. Na rozdíl od skutečného štěpení trajektorie, zde předpokládáme vznik velkého množství částic, tj. obvykle položíme $m = 10^2$ až 10^3.

- Ukončení trajektorie
  Obvykle při modelování transportu nenecháme statistickou váhu libovolně zmenšovat, ale udáme pro ni limitní hodnotu ϵ, pod kterou se již nevyplatí částici sledovat a trajektorii dále modelovat. Můžeme však trajektorii opustit i přesnějším způsobem, inverzním k metodě rozdělení trajektorií. Bude-li váha částice příliš malá, rozehrajeme fiktivní srážku, při níž částice s pravděpodobností $1 - 1/m$ zaniká a s pravděpodobností $1/m$ zvětšuje svou váhu m-násobně.

- Systematický výběr
  V pracovní oblasti budeme chtít modelovat pohyb n částic rozdělených s hustotou $p(r)$ a na počátku se náhodně pohybujících. Chceme-li mít o určité části pracovní oblasti G přesnější informaci, rozdělíme ji na m vzájemně se nepřekrývajících částí $G = G_1 + G_2 + \cdots + G_m$ a do každé z nich dáme n_i částic. Těmto částicím přiřadíme v různých dílčích oblastech G_i různé váhy

  $$w_i = \frac{n}{n_i} \cdot \int_{G_i} p(r)\,\mathrm{d}r \qquad i = 1, \ldots, m\,.$$

  Dílčí počty částic mohou být libovolné (obecně $\sum n_i \neq n$), musí však platit podmínka $\sum\limits_i n_i\, w_i = n$.

- Metoda podobných trajektorií
  Máme-li skupinu úloh geometricky podobných, provedeme řešení pouze pro jednu z nich a v dalších úlohách trajektorie příslušně přetransformujeme. Znamená to změnit volné dráhy ξ a zavést váhové faktory částic závislé na změně měřítka problému.

- Efektivnější transformace diskrétní náhodné veličiny
  Pro rozehrávání diskrétní náhodné veličiny máme k dispozici standardní postup znázorněný na obr. 4.2. Tento postup je vhodný pro všechny problémy, může však být pro některé z nich neefektivní. Konkrétně v případě, když pracujeme s diskrétní náhodnou veličinou zadanou tabulkou

  $$\xi = \begin{pmatrix} x_1 & x_2 & \ldots & x_k \\ p_1 & p_2 & \ldots & p_k \end{pmatrix}$$

  s velkým počtem sloupců, postupné prohledávání všech dílčích intervalů $p_1, p_1+p_2, p_1+p_2+p_3, \ldots$ je velmi pomalé. Prohledávání proto můžeme zkrátit různými postupy – např. metodou půlení intervalu (viz Problémy k řešení na konci této kapitoly) nebo i přímým nalezením konkrétního intervalu pomocí tzv. algoritmu Nanbu [13].

4.5.3 Jiné fyzikální problémy

Metoda Monte Carlo se liší od obvyklých numerických metod i jiných algoritmů počítačové fyziky v tom, že neposkytuje nějaký jednoduchý vzorec nebo návod na řešení problému. Při této metodice počítačového experimentu je algoritmus téměř každého řešeného problému originální. To platí i pro řešení transportních problémů, jejichž modifikace jsou ve fyzice nejrozšířenější. Proto vztahy a postupy v této sekci uvedené netvoří jednoznačný návod, ale je třeba je chápat jen jako „stavební kameny“, z nichž si každý musí zkonstruovat svůj vlastní algoritmus.

Metoda stochastického počítačového experimentu ve fyzice se ovšem neomezuje jen na řešení transportních problémů. Další oblasti použití jsou však tak rozdílné, že je nelze převést na společný základ a na obecné úrovni vyložit. Naším problémem může být např. určení pravděpodobnostního chování složitých systémů, které lze jen obtížně analyzovat přímo. V metodě Monte Carlo místo toho postačí mít detailní informaci o jednotlivých prvcích systému a chování celého systému již z toho odvodíme.

Jako příklad řešení problému tohoto typu si vezměme analýzu složitého elektronického obvodu nebo přístroje. Jeho zapojení se skládá z velkého množství aktivních i pasívních prvků – odporů, kondenzátorů, tranzistorů, atd. (na podobném principu probíhá i studium integrovaných obvodů, které však svou složitostí překračují rámec tohoto studijního textu). Fyzici obvykle řeší tzv. analýzu elektronického obvodu, kdy na základě znalosti konkrétního elektrického schématu a hodnot jednotlivých jeho prvků počítají nějakou charakteristiku popisující stav celé soustavy – např. zesílení nebo napětí a proud v koncovém stupni. Před inženýry a konstruktéry takových přístrojů ale stojí složitější problémy. Jedná se především o tzv. syntézu obvodu, kdy jsou zadány požadované výsledné charakteristiky celého obvodu nebo přístroje a je třeba navrhnout rozmístění součástek a jejich hodnoty. Při výrobě elektronických zařízení k tomu přispívá ještě další problém přenést tyto teoretické výpočty do praxe. Při montáži přístroje se projeví to, že každý prvek zapojení má hodnotu poněkud se lišící od nominální, což se projeví jako rozptyl výsledné charakteristiky přístroje. Dříve proto standardním postupem bylo vyrobit větší počet požadovaných zařízení, ta detailně proměřit a najít rozptyl výsledných charakteristik. Pokud byl rozptyl v požadované toleranci, bylo možno přistoupit k výrobě, jinak se zařízení vracelo do konstrukce k přepracování zapojení.

Totéž lze s větší efektivitou provést i metodou Monte Carlo. Jako vstupní hodnoty pro modelování potřebujeme rozdělovací funkce hodnot jednotlivých prvků zapojení, pak rozehrajeme jednotlivé hodnoty všech prvků a na základě elektrického schématu spočítáme výslednou charakteristiku přístroje.

Tento postup je ekvivalentní transportnímu problému, ale místo rozptylových procesů budeme mít charakteristiky prvků zapojení, místo jedné trajektorie budeme mít obvod osazený jednou sadou nagenerovaných hodnot prvků a opakování výpočtů a statistická analýza výsledků je stejná. Podobně jako jsme při transportu částic využívali vlastnosti náhodných volných drah a dostali vztah (4.45), zde budeme předpokládat, že odchylky hodnot jednotlivých elektronických prvků jsou rozdělené podle Gaussovy rozdělovací funkce. Potom můžeme přímo převzít údaje udávané výrobcem součástech a vhodně je interpretovat – např. hodnotu odporu rezistoru $R = 1000\ \Omega \pm 5\ \%$ budeme interpretovat tak, že odpor souboru rezistorů je rozdělen podle Gaussovy křivky s parametry $N(1000, 16.667)$, tj. nominální hodnota 1000 přímo určuje první parametr μ a druhý parametr σ se určí podle známého pravidla tří sigma z hodnoty $1000 \cdot 0,05$.

Dalším problémem, kde se s úspěchem využívá metoda Monte Carlo, je vyhodnocování životnosti různých složitých zařízení. Klasický postup při výrobě těchto zařízení vycházel z detailního proměření určité sady přístrojů a podle jejich jednotlivých životností byla spočítána průměrná životnost a její rozptyl, což se srovnávalo se záruční dobou a podle toho buď vývoj pokračoval a nebo se již zahájila výroba. Při použití metody Monte Carlo je třeba mít jako vstupní hodnoty životnosti jednotlivých uzlů zařízení i s příslušnými tolerancemi (opět s předpokladem Gaussova rozdělení). Pak se provede výpočet průměrné životnosti celého zařízení jako opakovaný výpočet životnosti jednotlivých nagenerovaných sad součástek. Přitom se musí brát ohled na řazení jednotlivých prvků a uzlů v přístroji – má-li větev prvky řazené sériově, je životnost celé větve udána minimální životností prvků větve, naproti tomu při paralelním zapojení životností maximální. Při řešení tohoto problému se vychází z algoritmů formálně podobných předchozím případům, tj. rozehrání trajektorie v transportním problému nebo řešení Kirchhoffových zákonů v elektrických obvodech.

4.6 Shrnutí

V této kapitole bylo již uvedeno větší množství poznatků, které je nutno úspěšně zvládnout, chceme-li umět stochasticky částicově modelovat. Pokud si to stručně shrneme, jedná o tyto okruhy znalostí:

- **Základní schéma metody Monte Carlo**
- **Základní pojmy počtu pravděpodobnosti a matematické statistiky – náhodné veličiny a jejich popis, charakteristiky náhodných veličin, vybrané věty**

- **Metody generování náhodných čísel**
- **Metody transformace diskrétních a spojitých náhodných veličin včetně příkladů transformace vybraných veličin potřebných ve fyzice**
- **Vybrané algoritmy řešení problémů numerické matematiky metodou Monte Carlo – numerická integrace a řešení Laplaceovy rovnice**
- **Transportní problém ve fyzice – princip, náhodná volná dráha, srážkové procesy, štěpení trajektorie**
- **Umělé obraty v metodě Monte Carlo.**

4.7 Problémy k řešení

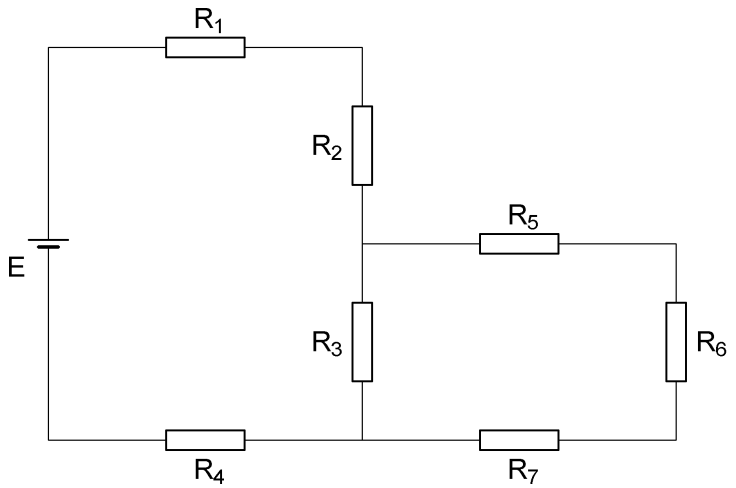
Problémy určené k procvičování látky popsané v této kapitole buď formou samostudia nebo v rámci organizované výuky:

1. Odvoďte vztah pro určení hodnoty π metodou Buffonovy jehly (Výsledek: $\pi \doteq 2\,\frac{l/d}{M/N}$).
2. Zkuste navrhnout další algoritmy pro určení hodnoty konstanty π pomocí metody Monte Carlo.
3. Naprogramujte a otestujte některé generátory pseudonáhodných čísel uvedené v sekci 4.3.1.
4. Napište programy na numerickou integraci funkce jedné proměnné dvěma alternativními postupy – algoritmem numerické matematiky a metodou Monte Carlo. V obou případech optimalizujte oba programy na rychlost s využitím poznatků obou disciplín a srovnejte efektivitu obou přístupů.
5. Napište programy na numerickou integraci funkce dvou proměnných opět dvěma alternativními postupy – algoritmem numerické matematiky a metodou Monte Carlo. Srovnejte efektivitu obou přístupů.
6. Naprogramujte transport částic prostředím obsahujícím pružné srážky, nepružné srážky a záchyt s cílem určit koeficient záchytu daného prostředí ve tvaru rovinné desky. Hodnoty tloušťky desky, středních volných drah pro jednotlivé typy interakcí, ztrátu energie při nepružné

srážce ΔE a počet vystupujících částic volte jako parametry modelu nastavitelné pomocí vstupního menu programu. U pružných i nepružných interakcí předpokládejte úhlově izotropní rozptyl. Výpočet proveďte přirozenou metodou Monte Carlo a sledujte, jak se doba výpočtu mění se zkracující se střední volnou drahou jednotlivých rozptylových procesů a zejména jak narůstá se vzrůstem intenzity záchytu.

7. Proveďte stejný výpočet modelem s umělým obratem nahrazujícím záchyt a srovnejte vzrůst efektivity simulace.

8. Vyřešte elektrické zapojení znázorněné na obr. 4.14 s cílem spočítat napětí na rezistoru R_6, tj. veličinu U_6, pro ideální hodnoty součástek: $E = 10\,V, R_1 = 10\,\Omega, R_2 = 10\,\Omega, R_3 = 150\,\Omega, R_4 = 5\,\Omega, R_5 = 25\,\Omega, R_6 = 100\,\Omega, R_7 = 25\,\Omega$ (Výsledek: $U_6 = 5\,V$).



Obrázek 4.14: Elektrické schéma.

Pak pomocí metody Monte Carlo spočtěte rozptyl hodnot výstupního napětí U_6 pro rozptyl hodnot rezistorů:
a) R_1 až R_7: $\pm 5\%$,
b) R_1 až R_7: $\pm 1\%$,
c) R_1 až R_5 a R_7: $\pm 5\%$, R_6: $\pm 1\%$.

9. Napište program na generování Maxwellova rozdělení částic.
Tento program je jako vzorový naprogramován dvěma způsoby a jeho výpis spolu s výstupem z programu je uveden v dalším paragrafu.

10. Stochasticky namodelujte 2D růst tenkých vrstev – popis řešení viz další paragraf.

Vyřešený problém č. 9: Generování Maxwellova rozdělení

Maxwellovo-Bolzmannovo rozdělení patří mezi základní rozdělovací funkce ve fyzice spolu s rozděleními Fermiho-Diraca a Boseho-Einsteina. Toto rozdělení mají klasické částice jako jsou atomy plynu, elektrony a ionty v plazmatu, apod., proto se v počítačové fyzice velmi často používá. Maxwellova rozdělovací funkce (4.15) má obecný tvar

$$p(x) = \frac{2}{a^3\sqrt{2\pi}} \cdot x^2 \cdot \exp\left(-\frac{x^2}{2\,a^2}\right) \qquad x \in \langle 0, \infty) \,,$$

který v terminologii kinetické teorie plynů přechází na výraz

$$p(x) = \frac{2}{\sqrt{2\pi}} \cdot \left(\frac{m}{kT_B}\right)^{3/2} \cdot v^2 \cdot \exp\left(-\frac{m\,v^2}{2\,k_B T}\right).$$

Zde v je rychlost částice ležící v intervalu $\langle 0, \infty)$, m je hmotnost částice, k_B Boltzmannova konstanta a T teplota souboru částic.

Na hustotu pravděpodobnosti Maxwellova rozdělení nelze použít metodu inverzní funkce (4.26), neboť integrál typu $\int x^2\, e^{-x^2} \mathrm{d}x$ není analyticky řešitelný stejně jako v případě Gaussova rozdělení $\int e^{-x^2} \mathrm{d}x$. V literatuře proto bylo navrženo několik přibližných postupů, z nichž si dva uvedeme a srovnáme.

První algoritmus je založený na tabelované distribuční funkci Maxwellova rozdělení $F(v)$. V proceduře *MAXWELL1_SEED* vektor $F(v)$ naplníme celkem 1 000 hodnotami distribuční funkce. Při volání výkonné procedury *MAXWELL1* vždy dostaneme jednu hodnotu rychlosti v s maxwellovským rozdělením. Jedná se o příklad transformace diskrétní náhodné veličiny. Pro urychlení prohledávání tak dlouhého vektoru se používá metoda půlení intervalu, která zaručuje nalezení správné hodnoty z tabulky F v deseti krocích. Pro zpřesnění výpočtu je ještě prováděna v tabulce lineární interpolace.

Druhý algoritmus naprogramovaný v proceduře *MAXWELL2* je založen na vztahu typu $\xi = g(\gamma_1, \ldots, \gamma_n)$. Pro Maxwellovo rozdělení byl nalezen přibližný vzorec

$$v = \sqrt{-\ln\gamma_1 - \ln\gamma_2 \cdot \cos\left(2\,\pi\,\gamma_3\right)^2} \,.$$

Oba algoritmy jsou upraveny tak, aby generovaly částice s maxwellovským rozdělením rychlosti normalizovaným na rozdělení

$$N(v) = C \cdot v^2 \cdot exp(-0.00002 \cdot v^2) \,.$$

Převod na reálné fyzikální parametry m, T a k_B je proveden v konstantě v_{Max}, kterou se získaná rychlost vynásobí. V programu je rozehrávána

pouze celková rychlost částic v. Pokud bude třeba generovat i složky rychlosti, příslušné hodnoty (v_x, v_y, v_z) se získají pomocí směrových kosínů (4.30).

Výpis programu v programovacím jazyku FORTRAN 90:

```
Program MAXWELL

! Generovani Maxwellova rozdeleni rychlosti

!    Castice: atomy Ar
!    Vystup - rychlosti castic: v [m/s]
!    Programovací jazyk Compaq FORTRAN 6.6

   use DFLIB
   implicit none
   integer(4)  nMax,Steps
   real(4)     kB,m,T,pi
   parameter                          &
      (kB    = 1.38062E-23,           & ! Boltzmannova konstanta
       nMax  = 1000000,               & ! celkovy pocet castic
       m     = 6.68173E-26,           & ! hmotnost castic [kg]
       T     = 300.0,                 & ! 'teplota' castic [K]
       pi    = 3.14159265,            & ! konstanta pi
       Steps = 1000)                    ! pocet kroku tabulky F
   integer(4)  i,n,j
   real(4)     gama,v1,v2,v1Mean,v2Mean,vMax
   integer(4)  vMaxwell(2,50)
   real(8)     F(0:Steps)

! inicializace

   ! maximalni rozsah tabulky rychlosti
   vMax=5.0*sqrt(2.0*kB*T/m)
   ! inicializace generatoru nahodnych cisel
   call SEED(RND$TIMESEED)
   ! inicializace generatoru Maxwellova rozdeleni
   call MAXWELL1_SEED(F)
```

```
! generovani rychlostniho rozdeleni castic obema generatory

   v1Mean=0.0
   v2Mean=0.0
   vMaxwell=0.0
   do i=1,nMax
      call MAXWELL1(F,v1,vMax)
      v1Mean=v1Mean+v1
      j=int(50.0*v1/vMax)+1
      vMaxwell(1,j)=vMaxwell(1,j)+1
      call MAXWELL2(v2,vMax)
      v2Mean=v2Mean+v2
      j=int(50.0*v2/vMax)+1
      vMaxwell(2,j)=vMaxwell(2,j)+1
   end do

! testovani generatoru

   open(6,file="Results.txt",access='SEQUENTIAL',status='UNKNOWN')
      write(6,100)
      write(6,101)
      write(6,103) v1Mean/nMax,v2Mean/nMax
      write(6,102)
      write(6,104)
      do j=1,50
         write(6,105) j,vMaxwell(1,j),vMaxwell(2,j)
      end do
   close(6)

 100 format("SROVNAVACI TEST GENERATORU MAXWELLOVA ROZDELENI 1 A 2"/)
 101 format("a) Stredni hodnoty:"/)
 102 format("b) Rozdeleni celkove rychlosti:"/)
 103 format("   vMean  =",F10.1,F14.1/)
 104 format("    v         N1(v)N2(v)"/)
 105 format(I5,2I14)

   stop
end

!=====================================================================
```

```
Subroutine MAXWELL1_SEED(F)

! Priprava generovani Maxwellova rozdeleni rychlosti

   implicit none
   integer(4),parameter :: Steps = 1000     ! pocet kroku tabulky F
   integer(4)              i
   real(8)                 F(0:Steps),norm,v

   norm=0.0
   do i=1,Steps
      v=i-0.5
      F(i)=v**2*exp(-0.00002*v**2)
      norm=norm+F(i)
   end do
   F(0)=0D0
   do i=1,Steps
      F(i)=F(i-1)+F(i)/norm
   end do
   F(Steps)=1D0

   return
end subroutine MAXWELL1_SEED

!---------------------------------------------------------------------

Subroutine MAXWELL1(F,v,vMax)

! Generovani Maxwellova rozdeleni castic - metoda 1

   implicit none
   integer(4),parameter :: Steps = 1000        ! pocet kroku tabulky F
   real(4),parameter ::    pi    = 3.14159265  ! konstanta pi
   integer(2)              j1,j2,j3
   real(4)                 v,gama,vMax
   real(8)                 F(0:Steps)
```

```
   ! generovani celkove rychlosti v

   call RANDOM(gama)
   j1=0
   j2=Steps
1  j3=(j1+j2)/2
   if (gama.lt.F(j3)) then
      j2=j3
   else
      j1=j3
   end if
   if ((j2-j1).gt.1) goto 1
   call RANDOM(gama)
   v=(j2-gama)*vMax/Steps

   return
end subroutine MAXWELL1

!---------------------------------------------------------------------

Subroutine MAXWELL2(v,vMax)

! Generovani Maxwellova rozdeleni castic - metoda 2

   implicit none
   real(4),parameter :: pi = 3.14159265     ! konstanta pi
   real(4)              gama1,gama2,gama3,v,vMax

   call RANDOM(gama1)
   call RANDOM(gama2)
   call RANDOM(gama3)
   v=0.125*sqrt(pi)*sqrt(-log(gama1)-log(gama2)*cos(2.0*pi*gama3)**2)*vMax

   return
end subroutine MAXWELL2

!=====================================================================
```

Výstup z programu:

```
SROVNAVACI TEST GENERATORU MAXWELLOVA ROZDELENI 1 A 2

a) Stredni hodnoty:

   vMean  =     443.9         440.1

b) Rozdeleni celkove rychlosti:

    v          N1(v)          N2(v)

    1            530            574
    2           3738           3757
    3           9697           9773
    4          18146          18578
    5          27954          28489
    6          38588          38979
    7          48917          49384
    8          57827          59254
    9          65591          66551
   10          70725          71468
   11          73835          74932
   12          73639          74837
   13          72370          72488
   14          67841          68298
   15          63127          62822
   16          56245          56420
   17          50263          49092
   18          42484          42077
   19          35664          35322
   20          29179          28485
   21          23507          22637
   22          18419          17842
   23          14175          13471
   24          10844          10017
   25           8026           7520
   26           5877           5398
   27           4092           3770
   28           2943           2646
   29           1982           1799
```

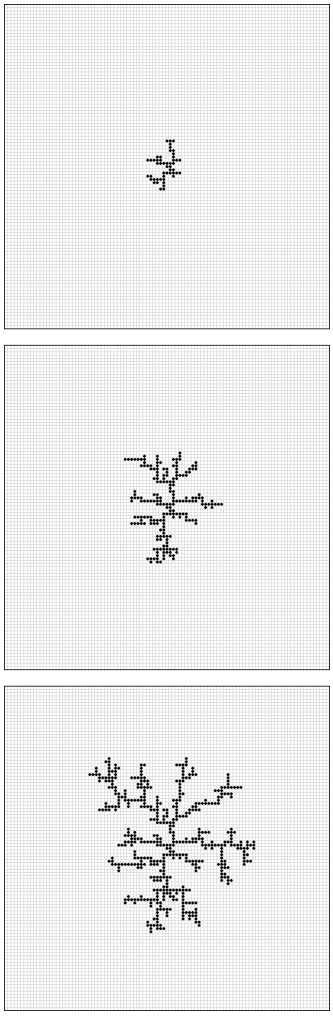
30	1383	1209
31	876	788
32	573	492
33	354	347
34	237	192
35	148	130
36	80	66
37	53	47
38	30	19
39	12	11
40	14	12
41	7	1
42	5	2
43	2	0
44	1	3
45	0	1
46	0	0
47	0	0
48	0	0
49	0	0
50	0	0

Vyřešený problém č. 10: 2D růst tenkých vrstev

Jednou z metod nanášení tenkých vrstev různých materiálů je jejich postupné vytváření dopadem atomů na podložku za sníženého tlaku [14]. Pokud celý proces neprobíhá při teplotě absolutní nuly, atom po dopadu na podložku po ní určitou dobu migruje. Vedle řady dalších procesů se zřejmě největší měrou uplatňuje mechanismus růstu, při kterém se migrující atom dostane do sousedství dalšího atomu nebo skupiny atomů nanášené látky, což migraci zbrzdí nebo zcela zastaví a vzniká stabilní útvar. Podle konkrétní kombinace atomů nanášené látky a atomů podložky (tj. podle poměru jejich vazebních energií) může být vznikající útvar dvourozměrný (2D - kdy nanášené atomy nejprve zaplňují povrch podložky v jedné rovině a až teprve po vytvoření tzv. monovrstvy začnou zaplňovat rovinu další) nebo třírozměrný (3D - vznikající objekty mají přibližně tvar kulových vrchlíků).

Modelování úplného procesu růstu tenkých vrstev včetně započítání reálných vlastností povrchu a vznikajících objektů patří k velmi složitým problémům počítačové fyziky řešitelným pouze kombinací několika přístupů a překračuje to jak bakalářskou tak i magisterskou úroveň výuky. Pokud však si dáme značně omezující předpoklady, lze kvalitativní model tohoto procesu vytvořit i pomocí metody Monte Carlo.

Budeme předpokládat, že atomy dopadají na podložku náhodně a po podložce náhodně migrují. Migrace může být ukončena dvěma způsoby – buď atom dospěje k již existujícímu útvaru, k němuž se připojí, a nebo po určité době opět podložku opustí. Tento druhý proces zjednodušeně naprogramujeme tak, že migrující atom přestaneme sledovat po opuštění pracovní plochy. Migrace je náhodný proces a proto je přirozené na jeho simulování použít metodu Monte Carlo. Budeme zjednodušeně předpokládat, že podložka představuje čtvercovou síť (v programu realizovanou dvourozměrným polem) a atom při migraci náhodně přeskakuje mezi sousedními uzly této sítě, tj. atom má v každé pozici 25-procentní pravděpodobnost pohybu v jednom ze čtyř na sebe kolmých směrů. Dále budeme předpokládat, že migrace probíhá v pravidelných krocích, při nichž se vždy částice přemístí o jeden uzel sítě. Pokud nás bude zajímat jen tvar vznikajícího objektu, můžeme simulaci provádět jednočásticově – do pracovní oblasti vždy vypustit pouze jeden atom a modelovat jeho trajektorii až do konce (připojením k existujícímu útvaru nebo opuštěním pracovní plochy) a teprve pak vypustit další atom a proces opakovat. Dále budeme předpokládat, že atom, který se přiblížil k objektu, se k němu připojí a již se dále vůbec pohybovat nebude. To vše jsou zjednodušující předpoklady, fyzikálně sice možné, ale platné jen v dosti výjimečných případech. Při jejich použití však je model i vytvořený program relativně jednoduchý a povede na výsledky podobné obr. 4.15.



Obrázek 4.15: Model 2D růstu vrstev. Počet atomů: 50, 200 a 500.

Praktické poznámky k vytváření programu:

1. Umístění rostoucího objektu ve středu pracovní oblasti
   Růst objektu je nutno nastartovat na předem umístěném kondenzačním jádru. Toto jádro ve tvaru jednoho nepohyblivého atomu umístíme do středu podložky.

2. Dodržení náhodnosti migrace
   Při skutečném procesu vytváření vrstvy dopadají atomy na podložku v širokém okolí rostoucího objektu a migrují k němu rovnoměrně ze všech stran. Čtvercový rozměr pracovní oblasti není pro úhlovou izotropnost migrace vhodný, proto je třeba vytvořit kružnici, z níž budou

atomy vypouštěny. Pokud budeme modelovat objekt tvořený velkým počtem atomů, je třeba zvolit generační kružnici velkou, aby byl objekt i v konečné fázi růstu od ní dosti vzdálen, jinak je náhodnost růstu porušena.

3. Zvýšení efektivity programu
   Při náhodné migraci mohou atomy migrovat nejen směrem ke středu podložky, ale i k jejímu okraji a mimo ni. Abychom zamezili vzniku nekonečného cyklu v programu, je třeba tyto atomy vyloučit. Je však ale možné, že při migraci atomy občas (zejména na začátku migrace) opustí původní generační kružnici, ale pak se ke středu podložky vrátí a k rostoucímu objektu přispějí. Automatické ukončení trajektorie po překročení generační kružnice by mohlo náhodnost růstu i efektivitu programu snížit. Rozumným kompromisem proto je vytvořit druhou kružnici o větším poloměru a teprve po jejím překročení trajektorii ukončit.
   Pokud je generační kružnice příliš velká, růst z počátku probíhá příliš pomalu – malý objekt představuje příliš malý cíl pro záchyt migrujícího atomu. Optimalizovaný program proto začíná s dosti malou generační kružnicí (a též kružnicí, kde trajektorie případně končí) a v průběhu programu spolu s růstem objektu zvětšuje i poloměry obou kružnic.

4. Tvar vzniklého objektu
   Jak je z obrázku zřejmé, modelováním 2D objektu pomocí popsaného jednoduchého algoritmu metodou Monte Carlo dostaneme fraktální útvar, jehož vlastnosti můžeme dále studovat. Charakter tohoto fraktalu můžeme ovlivňovat migračními podmínkami v modelu – tvarem sítě tvořící podložku, umožněním omezené migrace atomů i po připojení k objektu, atd.

Kapitola 5

METODA MOLEKULÁRNÍ DYNAMIKY

5.1 Princip metody

Metoda molekulární dynamiky patří mezi metody deterministického částicového modelování. Je to metoda *částicová*, což znamená, že studovaný jev popisujeme na základě chování souboru jeho dílčích částí. Je to metoda *deterministická*, při níž řešíme klasické pohybové rovnice (existují i metody kvantové). Částice, jejichž chování budeme metodou molekulární dynamiky simulovat, spolu vzájemně interagují, proto se bude jednat o metodu *mnohočásticovou* a příslušný datový soubor (viz obr. 4.7) bývá dosti rozsáhlý a je třeba s ukládáním dat zacházet úsporně.

Metoda vznikla v první polovině 50. let. Mezi první oblasti použití patřilo studium chování kapalin, později i pevných látek a dalších interagujících systémů. V současné době je tato metoda s úspěchem používána pro velmi rozmanité spektrum problémů od studia atomů až po astronomii a kosmologii. Terminologii však tato metoda převzala z fyziky kapalin, proto se můžeme občas setkat s označením pro částice ve studovaném souboru „molekuly“, což může představovat např. elektrony a ionty v plazmatu, atomy v plynech, hvězdy při studiu galaxií a galaxie při studiu metagalaxie. Rozhodující je, že částice, jejichž chování pomocí pohybových rovnic popisujeme, představují relativně mikroskopickou úroveň studovaného jevu.

Typické problémy, které se metodou molekulární dynamiky řeší, jsou modelování trajektorií „molekul“ (částic, těles). Obecný postup řešení takového problému je následující:

– Vytvoříme co nejvěrnější model studovaného jevu.

– Systém popíšeme pomocí souboru N částic.

– Sestavíme klasické pohybové rovnice pro všechny částice.
– Určíme počáteční podmínky a pohybové rovnice vyřešíme. Při tomto řešení obvykle pracujeme v rozmezí časů $\langle t_0, t_{max}\rangle$ a naším cílem je nalézt trajektorii částice nebo částic v tomto intervalu.

Metoda molekulární dynamiky je přirozenou metodou pro studium dynamiky částic nebo těles. V některých oblastech fyziky se však používá i pro výpočet statických vlastností. Nechť je třeba určit hodnotu nějaké fyzikální veličiny A, která je funkcí stavů $\vec{x} = (x_1, \ldots, x_N)$ s rozdělovací funkcí f. V tomto případě systém popíšeme pomocí souboru stavů $\vec{x}$. Naším cílem bude nalézt průměrnou hodnotu veličiny A přes soubor

$$< A >= \frac{\int_\Omega A(\vec{x})\, f(H(\vec{x}))\, \mathrm{d}\vec{x}}{\int_\Omega f(H(\vec{x}))\, \mathrm{d}\vec{x}}\,,$$

kde H je Hamiltonián modelu a Ω fázový prostor tvořený stavy $\vec{x}$.

Při simulaci nelze $< A >$ přímo určit. Veličinu A proto počítáme podél dráhy ve fázovém prostoru a určujeme časový průměr

$$\bar{A}_t = \frac{1}{t - t_0} \cdot \int_{t_0}^{t} A(\vec{x}(\tau))\, \mathrm{d}\tau\,.$$

Na základě ergodicity je známo, že lze nahradit průměr přes soubor časovým průměrem $< A >= \bar{A}_\infty$. V našem případě ale se dopouštíme dvou nepřesností:

– dráhu částic ve fázovém prostoru můžeme sledovat jen po konečnou dobu t, proto přibližně položíme $\bar{A}_t \simeq < A >$,
– můžeme pracovat jen se systémy konečné velikosti.

Pozn.: Konečná velikost systémů má i své výhody. Vlivem omezeného počtu částic/stavů studované veličiny fluktuují kolem svých rovnovážných hodnot a v těchto fluktuacích je uložena další fyzikální informace. Fluktuace primárních veličin závisejí na vlastnostech studovaného systému a lze z nich proto vypočítat termodynamické veličiny druhého řádu. Tato možnost má obecnou platnost – i v experimentální fyzice lze ze šumu elektrického proudu procházejího systémy s malým počtem atomů určit teplotu těchto objektů, apod.

5.2 Použití metody molekulární dynamiky

Při praktickém řešení fyzikálního problému pomocí metody molekulární dynamiky je třeba zvolit pracovní oblast, najít všechny síly působící na jednotlivé částice souboru, určit počáteční stav částic, numericky vyřešit příslušné pohybové rovnice všech částic a případně statisticky zpracovat získané výsledné trajektorie.

5.2.1 Pracovní oblast

Při volbě pracovní oblasti musíme najít odpověď na několik otázek – volba tvaru pracovní oblasti, velikosti pracovní oblasti a okrajových podmínek. Tyto odpovědi se budou lišit podle typu řešeného problému. Rozlišujeme dva základní případy:

– Studujeme pohyb těles nebo částic v omezené oblasti (např. trajektorie komet ve Sluneční soustavě).
  V tomto případě zvolíme pracovní oblast tak velkou, aby studovaná soustava byla celá umístěna v této oblasti.

– Studujeme chování velkého počtu částic zaujímajících velký objem.
  Pracovní oblast jež musí mít měřítko podle velikostí částic, bude proto mnohem menší než celý objem prostoru naplněného částicemi a zvolíme ji jako jakési „okno“ do studované soustavy částic.

V prvním případě nebudeme mít s odpovědí na naše otázky žádné obtíže – tvar a velikost oblasti přizpůsobíme studované soustavě a okrajové podmínky nemusíme formulovat, jelikož částice nemohou obvykle pracovní oblast opustit. Pokud tento případ výjimečně nastane (např. při modelování trajektorie sondy vypuštěné ze Země k vnějším planetám Sluneční soustavy, kolem kterých má pouze proletět a vyslat zpět snímky), překročení hranice pracovní oblasti signalizuje ukončení programu.

Tvar a další parametry pracovní oblasti proto musíme určovat v případě, kdy pracovní oblast tvoří jen malou část prostoru zaplněného částicemi – např. při modelování procesů v plynu, studiu pozemského i kosmického plazmatu, apod.

1. Tvar pracovní oblasti
   Při volbě tvaru pracovní oblasti se budeme řídit symetrií řešeného problému. Pokud jsou částice rozloženy pravidelně, podřídíme i tvar pracovní oblasti tomuto rozložení, neboť pak se zjednoduší popis procesů na hranici. Příkladem může být studium pohybu elektronů v pevné látce, kdy tvar pracovní oblasti obvykle napodobuje zvětšenou elementární buňku krystalové mřížky.

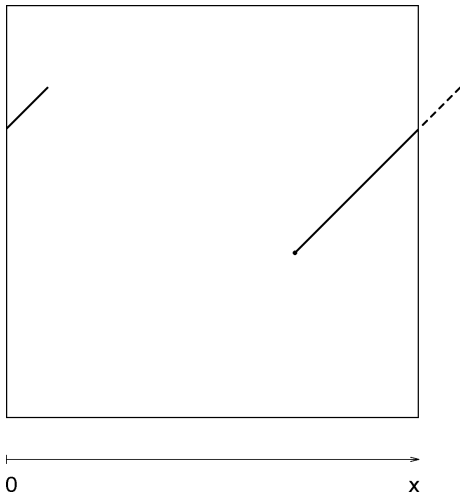
   Naopak, pokud studovaný problém žádnou výraznou symetrii nemá, volíme tvar pracovní oblasti co nejjednodušší, abychom si zjednodušili další manipulaci s daty při modelování. Ve dvourozměrném případě proto budeme pracovat se čtvercem o hraně L, ve třech rozměrech budeme volit pracovní oblast ve tvaru krychle (o hraně L). Zda pracovat v jednom, dvou nebo třech rozměrech, to musíme volit při formulaci modelu na základě studovaného fyzikálního jevu.

2. Velikost pracovní oblasti
   Pro volbu velikosti pracovní oblasti máme dvě kritéria – první výpočetní a druhé fyzikální. Jelikož při modelování nejraději pracujeme s přirozenými měřítky všech veličin, neboť pak je interpretace získaných výsledků nejjednodušší, snažíme se volit velikost pracovní oblasti podle počtu částic, které by měla obsahovat. Současná výpočetní technika nám obvykle umožňuje pracovat s $N = 10^3$ až 10^6 částicemi. Při menším počtu částic než $1 \cdot 10^3$ jsou příliš velké fluktuace a při příliš velkém počtu částic je výpočet příliš pomalý a nemusí stačit ani kapacita paměti počítače – současné největší kosmologické výpočty na superpočítačích pracují přibližně s 10^8 částicemi-galaxiemi. Např. pokud budeme modelovat procesy v nízkoteplotním plazmatu obsahujícím $1 \cdot 10^{15}$ nabitých částic v m^3 a budeme chtít vytvářet třírozměrný model s $N = 1 \cdot 10^6$ částicemi, dostaneme pro velikost pracovní oblasti ve tvaru krychle hodnotu $L = 1 \cdot 10^{-3}$ m.

   Druhé – fyzikální – kritérium vyplyne z charakteru silového působení v našem modelu. Rozeznáváme tzv. síly dalekodosahové, kdy jejich velikost se vzdáleností ubývá jako r^{-2} (sem patří především síla gravitační a elektrostatická), a síly krátkodosahové, které mají závislost na vzdálenosti mnohem silnější (např. r^{-7} v pevných látkách, atd.). Při volbě velikosti pracovní oblasti budeme požadovat, aby na částici umístěnou v jejím středu bylo silové působení od částic mimo pracovní oblast zanedbatelné, tj. aby na vzdálenosti $L/2$ klesly síly na nevýznamnou hodnotu. Je zřejmé, že toto kritérium se uplatní především u dalekodosahových sil.

   Ve většině případů nám současná aplikace obou kritérií určí interval, z kterého můžeme rozměr pracovní oblasti zvolit. Někdy však se obě kritéria navzájem vylučují. Pak nezbude nic jiného, než použít ekvivalent umělých obratů z metody Monte Carlo a začít pracovat s tzv. *makročásticemi*. Např. při modelování vývoje naší Galaxie místo s $10^{10} \div 10^{11}$ hvězdami budeme pracovat pouze s $10^6 \div 10^7$ „makrohvězdami“ a pro zachování reálného gravitačního působení musíme hmotnost každého takového tělesa v odpovídajícím poměru zvýšit. Pokud bychom podobný obrat provedli i při modelování procesů v plazmatu, museli bychom ve stejném poměru zvýšit hmotnost i náboj částic, aby důležitý poměr e/m zůstal zachován. Interpretace získaných výsledků však nebude jednoduchá – některé veličiny budou odpovídat realitě zatímco jiné se budou muset přeškálovat (např. zmenšením počtu částic v modelu proti realitě se určitě změní jejich srážková frekvence).

3. Volba okrajových podmínek
   V případě, že pracovní oblast je pouze výřezem (okénkem) z celého objemu naplněného částicemi, bude neustále docházet k vylétání částic z pracovní oblasti a naopak vstupu částic z okolního prostoru. V modelu tento rovnovážný proces obvykle simulujeme pomocí tzv. cyklických okrajových podmínek – pokud částice pracovní oblast opustí, ihned vstoupí do pracovní oblasti zpět na odpovídajícím místě protilehlé stěny (viz obr. 5.1). Pokud budeme mít pracovní oblast prostého tvaru, cyklické okrajové podmínky budeme realizovat velmi jednoduše. V kubické pracovní oblasti o hraně L každá souřadnice x, y a z každé částice musí ležet v rozmezí $\langle 0, L\rangle$. V modelu potom místo abychom neustále sledovali polohu částice a testovali to, zda se nepokouší opustit pracovní oblast, sledujeme pouze hodnoty souřadnic a přičteme nebo odečteme hodnotu L k příslušné souřadnici, která z intervalu $\langle 0, L\rangle$ vybočí.

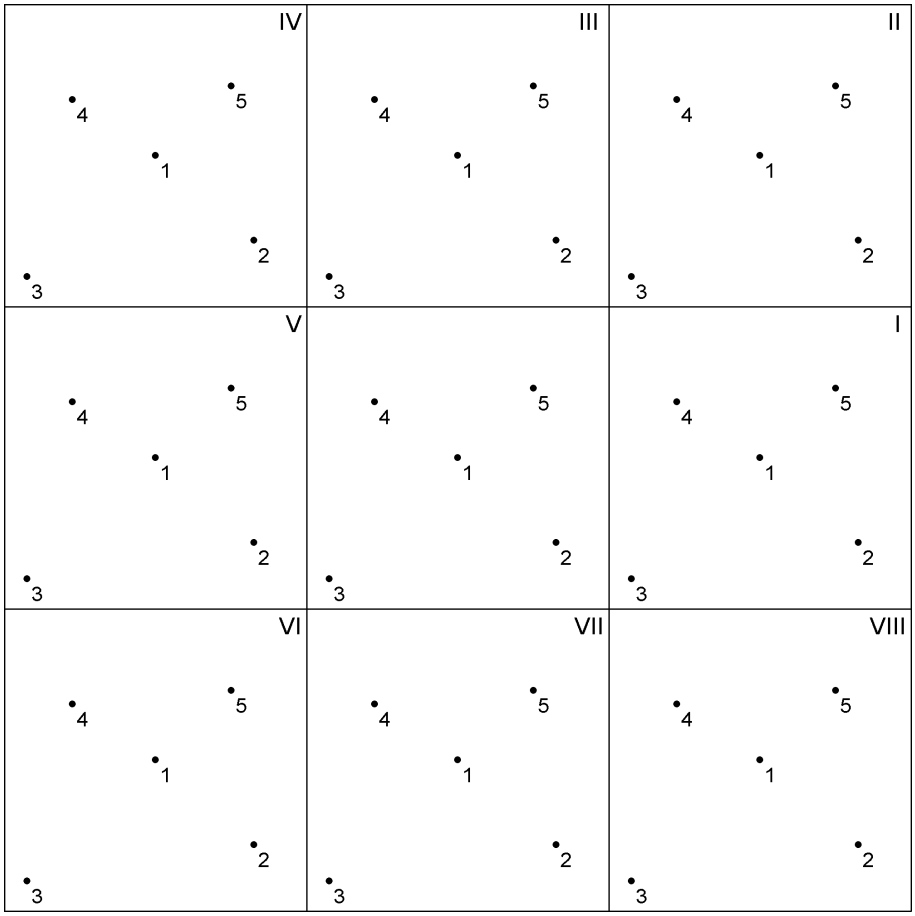


Obrázek 5.1: Cyklické okrajové podmínky ve směru osy x.

Při aplikaci cyklických okrajových podmínek si však musíme uvědomit, že tím na studovanou soustavu částic klademe určitá fyzikální omezení. V systému nesmí být přítomen žádný usměrněný tok částic (např. gradient teploty, atd.) a prostředí musí být zcela homogenní. Pokud tomu tak nebude, je nutno podobně jako při řešení transportního problému v metodě Monte Carlo v modelu vytvořit zdroj částic a jím kompenzovat úbytek částic opouštějících pracovní oblast.

5.2.2 Výpočet silového působení

Jelikož metoda molekulární dynamiky je založena na řešení pohybových rovnic, musíme stanovit síly, které na každou částici působí. Předpokládejme, že pracovní oblast obsahuje N částic. Sílu působící na částici nacházející se poblíž středu pracovní oblasti najdeme jako vektorový součet silového působení od ostatních $N - 1$ částic v pracovní oblasti.



Obrázek 5.2: Obrazy pracovní oblasti pro výpočet silového působení.

Jiná situace ale nastane pro částice ležící blíže k hranicím oblasti. Kdybychom započítali pouze silové působení od ostatních částic v pracovní oblasti, byl by výpočet síly chybný, neboť bychom zanedbali silnější silové působení od blízkých částic na druhé straně hranice pracovní oblasti. Proto je nutné nějakým způsobem do výpočtu síly započítat i jejich vliv. Nelze to však udělat přímo, neboť kdybychom do modelu zahrnuli vedle původních N

částic i některé částice sousední, znamenalo by to pouhé zvětšení pracovní oblasti a problém bychom jen přenesli k nové hranici.

Existuje několik metod řešení tohoto dilematu. Nejběžnější z nich je uvedena na obr. 5.2. V tomto obrázku je pracovní oblast znázorněna čtvercem uprostřed (bez dalšího označení). Oblast obsahuje N částic – v našem případě $N = 5$. Tuto pracovní oblast obklopíme ze všech stran kopiemi oblasti spolu s jejími částicemi – ve dvourozměrném modelu je těchto kopií osm, v třírozměrném 26. V našem obrázku jsou tyto kopie označeny čísly $I \div VIII$. Kopie zůstanou neustále svázané s původní oblastí, což znamená, že každý pohyb původní částice se promítne i do všech jejích obrazů.

Při nejrozšířenějším postupu se výsledné silové působení na konkrétní částici v původní pracovní oblasti i nadále počítá jako vektorový součet působení od ostatních $N - 1$ částic, přičemž se uvažují největší síly, tj. síly od nejbližších částic bez ohledu na jejich umístění. Konkrétně v našem případě dostaneme

$$\vec{F}_1 \;=\; \vec{f}_{12} + \vec{f}_{13} + \vec{f}_{14} + \vec{f}_{15}\,,$$

kde všechny částice 2, 3, 4 a 5 vezmeme v původní pracovní oblasti. Naproti tomu

$$\begin{aligned}
\vec{F}_2 &= \vec{f}_{21} + \vec{f}_{23}^{(I)} + \vec{f}_{24}^{(I)} + \vec{f}_{25}^{(VII)} \\
\vec{F}_3 &= \vec{f}_{31} + \vec{f}_{32}^{(V)} + \vec{f}_{34}^{(VII)} + \vec{f}_{35}^{(VI)} \\
\vec{F}_4 &= \vec{f}_{41} + \vec{f}_{42}^{(V)} + \vec{f}_{43}^{(III)} + \vec{f}_{45}^{(V)} \\
\vec{F}_5 &= \vec{f}_{51} + \vec{f}_{52}^{(III)} + \vec{f}_{53}^{(II)} + \vec{f}_{54}^{(I)}\,,
\end{aligned}$$

kde horním indexem rozlišujeme, zda částice patří do původní oblasti nebo některého jejího obrazu.

Při konkrétní realizaci tohoto postupu na počítači zjistíme, že nejbližších $N - 1$ částic k i-té částici má pořadová čísla $1, 2, \ldots, N$ (samozřejmě kromě i), neboli že každá částice nebo její obraz se v součtu popisujícím silové působení objeví právě jednou. To nám umožní algoritmus hledání největších sil zjednodušit. Za prvé, nebudeme pracovat přímo se silami ale pouze se vzdálenostmi částic, neboť všechny typy sil monotónně ubývají se vzdáleností a stačí proto najít $N - 1$ nejbližších částic. Za druhé, jelikož nepotřebujeme síly, nejedná se nám o absolutní hodnoty vzdáleností, ale jen o jejich pořadí, takže stačí pracovat s druhými mocninami vzdáleností, což výpočet dále zefektivní. Za třetí si povšimněme toho, že všechny souřadnice všech obrazů částice se liší od původních souřadnic pouze přičtením nebo odečtením hodnoty L v rozdílech typu $x_i - x_j$, což lze v cyklu jednoduše realizovat (aspoň pro krychlovou nebo čtvercovou pracovní oblast).

Vedle tohoto postupu se někdy do výpočtu výsledného silového působení na i-tou částici započítají síly od všech ostatních částic v původní pracovní oblasti i ve všech obrazech. Ve dvou rozměrech budeme proto muset skládat $9 \cdot N - 1$ sil, ve třech rozměrech dokonce $27 \cdot N - 1$ sil. Poněkud tím sice zpřesníme výpočet a zbavíme se nutnosti hledat $N-1$ největších sil, celkově to však bude znamenat značné zpomalení výpočtu. Totéž platí i pro další metody, kdy pracovní oblast obklopíme ne jednou ale dvěma vrstvami obrazů a skládáme síly od všech jejich částic (někdy se používá poněkud efektivnější postup, kdy skládáme síly od nekonečného počtu obrazů, neboť zde lze s úspěchem využít efektivní algoritmy na sčítání nekonečných řad).

Fyzikálně výše uvedený postup znamená, že soubor velkého počtu náhodně rozložených částic nahradíme souborem periodickým. Tato aproximace je zcela přesná pouze v případě modelování krystalů, kdy periodicita atomů je skutečná. V ostatních případech (atomy plynu, nabité částice v plazmatu, hvězdy, atd.) je tento postup oprávněný pouze tehdy, pokud počet částic v pracovní oblasti N je veliký.

5.2.3 Pohybové rovnice

Deterministická metoda molekulární dynamiky je založena na řešení soustavy pohybových rovnic pro všechny částice, tj. rovnic typu

$$\vec{F}_i = m_i \, \vec{a}_i \,, \qquad i = 1, \ldots, N \,.$$

Na základě postupů numerické matematiky převedeme tyto diferenciální rovnice druhého řádu ($\vec{a} = \frac{\mathrm{d}^2\vec{r}}{\mathrm{d}t^2}$) na dvojnásobný počet rovnic řádu prvního (v proměnných $\vec{r}$ a $\vec{v}$) a přejdeme na rovnice diferenční (spojitou časovou osu nahradíme diskrétní posloupností časů $t_0, t_1, \ldots, t_{max}$, kde $t_{k+1} - t_k = \Delta t$). Zvolíme počáteční podmínky pro polohu a rychlost všech částic v čase t_0 a dostaneme jednoduchý algoritmus pro řešení soustavy pohybových rovnic pro všech N částic:

- Počáteční podmínky: $\vec{r}_i^{\,0}, \vec{v}_i^{\,0}$
- Přechod z času t_0 do $t_1, \ldots$
- Přechod z času t_k do t_{k+1}

$$\begin{aligned} \vec{r}_i^{\,k+1} &= \vec{r}_i^{\,k} + \vec{v}_i^{\,k} \, \Delta t + \frac{1}{2m_i} \, \vec{F}_i^{\,k} \, \Delta t^2 \\ \vec{v}_i^{\,k+1} &= \vec{v}_i^{\,k} + \frac{1}{m_i} \, \vec{F}_i^{\,k} \, \Delta t \qquad\qquad i = 1, \ldots, N \qquad (5.1) \\ \vec{F}_i^{\,k+1} &= \cdots \,. \end{aligned}$$

Tento algoritmus odpovídá Eulerově metodě pro řešení obyčejných diferenciálních rovnic. Jeho slabinou je, že je pouze prvního řádu přesnosti v Δt.

V praxi se používají silnější algoritmy. Pokud požadujeme především přesnost řešení, použijeme jeden z kvalitních algoritmů, které nabízí numerická matematika – metody typu Rungeho-Kutty vyšších řádů nebo odpovídající metody typu prediktor-korektor. Tyto algoritmy bývají ale pomalé, proto jsou vhodné pro menší počet částic N – typickým příkladem jsou výpočty pohybu těles ve Sluneční soustavě.

Naproti tomu ve fyzice plazmatu, v kinetické teorii plynů, apod., kde máme velký počet částic a rozhodující je pouze chování jejich celého souboru, dáváme přednost méně přesným ale rychlejším algoritmům. Omezená přesnost výpočtu je kompenzována obvyklým následným průměrováním trajektorií s cílem získat makroskopické veličiny jako je tlak, elektrický proud, apod. Jako optimální kompromis se jeví algoritmy druhého řádu přesnosti odpovídající modifikovaným Eulerovým metodám. Obvykle se používají dva přibližně rovnocenné algoritmy – Verletova metoda a metoda 'leap-frog'.

Algoritmus Verletovy metody pro řešení pohybových rovnic:

- Počáteční podmínky: $\vec{r}_i^{\,0}, \vec{v}_i^{\,0}$
- Přechod z času t_0 do $t_1, \ldots$
- Přechod z času t_k do t_{k+1}

$$\begin{aligned} \vec{r}_i^{\,k+1} &= \vec{r}_i^{\,k} + \vec{v}_i^{\,k}\,\Delta t + \frac{1}{2m_i}\,\vec{F}_i^{\,k}\,\Delta t^2 \\ \vec{F}_i^{\,k+1} &= \cdots \qquad\qquad i = 1, \ldots, N \\ \vec{v}_i^{\,k+1} &= \vec{v}_i^{\,k} + \frac{1}{2m_i}\,(\vec{F}_i^{\,k} + \vec{F}_i^{\,k+1})\,\Delta t\,. \end{aligned} \tag{5.2}$$

Algoritmus 'leap-frog' metody pro řešení pohybových rovnic:

- Počáteční podmínky: $\vec{r}_i^{\,0}, \vec{v}_i^{\,1/2}$
- Přechod z času t_0 do $t_1, \ldots$
- Přechod z času t_k do t_{k+1}

$$\begin{aligned} \vec{r}_i^{\,k+1} &= \vec{r}_i^{\,k} + \vec{v}_i^{\,k+1/2}\,\Delta t + \frac{1}{2m_i}\,\vec{F}_i^{\,k}\,\Delta t^2 \\ \vec{F}_i^{\,k+1} &= \cdots \qquad\qquad i = 1, \ldots, N \\ \vec{v}_i^{\,k+3/2} &= \vec{v}_i^{\,k+1/2} + \frac{1}{m_i}\,\vec{F}_i^{\,k+1}\,\Delta t\,. \end{aligned} \tag{5.3}$$

Mezi třemi uvedenými metodami - Eulerovou (5.1), Verletovou (5.2) a 'leap-frog' (5.3) existují tyto rozdíly:

– Jediná Eulerova metoda (5.1) je univerzální. Ostatní dva algoritmy mají změněné pořadí výpočtu nových hodnot $\vec{F}_i$ a $\vec{v}_i$. Tato změna není pouze formální, ale přináší fyzikální omezení pro algoritmy (5.2) a (5.3) – jsou použitelné pouze pro řešení problémů, kde síla nezávisí na rychlosti částice. V praxi to znamená, že Verletův algoritmus i 'leap-frog' metodu můžeme použít pro síly gravitační povahy a elektrostatické, ne však pro úplnou Lorentzovu sílu. Pokud potřebujeme studovat pohyb nabitých částic v magnetickém poli, musíme použít buď základní pomalý Eulerův algoritmus (5.1) a nebo algoritmy vyvinuté zvláště pro tento případ.

– Verletův algoritmus (5.2) má jednu další slabinu. Ve vztahu pro výpočet rychlosti používáme současně starou a novou hodnotu síly, $\vec{F}_i^{\,k}$ a $\vec{F}_i^{\,k+1}$, což klade zvýšené nároky na kapacitu paměti počítače. Zatímco ve zbývajících algoritmech (5.1) a (5.3) nám pro uložení síly stačí jediný vektor, v kterém postupně vyměňujeme staré hodnoty za nové, v algoritmu Verletově potřebujeme současně vektory dva.

– Slabinou algoritmu 'leap-frog' naproti tomu je, že k nastartování potřebuje počáteční hodnoty polohy a rychlosti v různých časech, $\vec{r}_i^{\,0}$ a $\vec{v}_i^{\,1/2}$. Jelikož tyto hodnoty často nejsou z experimentů k dispozici, bývá nezbytné hodnoty rychlosti $\vec{v}_i^{\,1/2}$ z původních hodnot $\vec{v}_i^{\,0}$ dopočítat.

Pokud síla závisí pouze na poloze částice, což je obvyklá situace ve výpočtech astronomických a při řešení většiny problémů z fyziky plazmatu, jsou Verletův a 'leap-frog' algoritmus v podstatě rovnocenné.

5.2.4 Další otázky

Ačkoliv pohybové rovnice jsou v podstatě dosti jednoduché, při jejich použití narazíme na řadu problémů, které je nutno nejprve analyzovat a pokusit se je překonat nebo si být aspoň vědomi jejich existence.

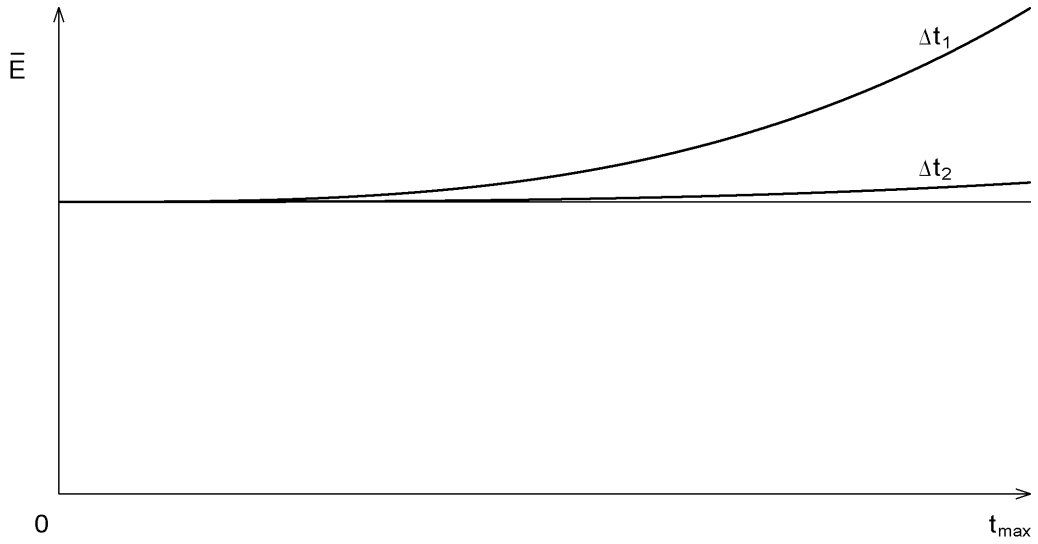
Prvním z nich je volba správné délky časového kroku Δt. Obecně lze prohlásit, že pokud je časový krok příliš velký, je výpočet trajektorie zatížený příliš velkou chybou. Naopak, pro příliš malý časový krok trvá výpočet příliš dlouho a může začít narůstat chyba zaokrouhlovací. Důležitější je ale fyzikální kritérium. Podle vzorkovacího teorému teorie informaci není možno získat informaci o procesech, které se odehrávají rychleji než odpovídá kroku Δt. Je proto nutno studovaný jev analyzovat z tohoto hlediska, najít nejvyšší frekvenci a jí přizpůsobit časový krok.

Pro některé fyzikální problémy může být situace mnohem komplikovanější. Existují systémy, které jsou podle výše uvedeného fyzikálního kritéria charakterizovány dvěma dosti rozdílnými časovými kroky a může být velmi obtížné sladit jejich požadavky. Např. plazma je tvořeno dvěma typy částic – lehkými rychlými elektrony a těžkými pomalými ionty. V podmínkách nízkoteplotního plazmatu při tlaku kolem 100 Pa získáme vhodné časové kroky pro elektrony a ionty $\Delta t_e \sim 10^{-12}s$ a $\Delta t_i \sim 10^{-8}s$, přičemž o ustálení celé soustavy částic rozhoduje její pomalejší složka. Důsledkem je potřeba integrovat soustavu pohybových rovnic v časovém intervalu $\langle t_0, t_{max}\rangle$ pro $t_{max} \sim 10^{-3}s$. Z toho pro elektrony vyplývá potřeba provést přibližně 10^9 kroků, což pro velký počet částic v modelu překračuje možnosti současné výpočetní techniky. Pro řešení tohoto problému nepostačí samotná metoda molekulární dynamiky a úloha se musí řešit kombinací několika postupů – viz kapitola 6.

Dalším možným problémem je nevhodná volba počátečních podmínek. Algoritmy (5.1) až (5.3) vyžadují zadání poloh a rychlostí všech částic na počátku výpočtu (tzv. fázi inicializace). Pokud tato data získáváme z experimentu (např. při modelování pohybu těles ve Sluneční soustavě), je vše v pořádku. V některých případech ale data musíme generovat počítačem na základě určitých obecných zákonitostí. To se týká všech problémů z oblasti mikrosvěta, kde polohy elektronů, iontů a atomů nám experiment není schopen poskytnout, ale i problémů astronomických, kde např. souřadnice hvězd v Galaxii nebo i v její menší části též nejsou známy. V tomto případě data nagenerujeme podle známé rozdělovací funkce, ale naplnění tohoto rozdělení je pro konečný počet částic jen přibližné – čím bude částic v modelu méně, tím může být chyba větší. Abychom případnou nepříznivou fluktuaci v prostorovém nebo rychlostním rozdělení částic co nejvíce potlačili, doporučuje se zařadit do procesu tzv. fázi ekvilibrace. Znamená to, že po počátečním rozmístění částic do pracovní oblasti nezačneme ihned provádět aktivní výpočet, ale po určitou dobu určujeme trajektorie částic aniž bychom zaznamenávali mezivýsledky (např. tlak, proud na sondu, apod.). Během této fáze výpočtu se případné nevhodné nastavení počátečních podmínek vykompenzuje a po jejím proběhnutí již můžeme výsledky získávat a ukládat (tomu se říká produkční fáze). Celkový výpočet se proto obecně skládá ze tří fází: fáze inicializace – fáze ekvilibrace – fáze produkční.

Pokud budeme důslední, ani tento problém nemůžeme zařadit pod metodu molekulární dynamiky, neboť pro generování poloh a rychlostí částic podle určitého teoretického rozdělení musíme použít postupy stochastické. Výsledný výpočet je proto prováděn hybridní technikou a jeho popis opět patří do kapitoly 6.

Konečně největším problémem je tzv. „nefyzikální ohřev“. Pokud vezmeme uzavřený objem naplněný částicemi (např. atomy ideálního plynu umístěné v nějaké nádobě) a držíme soustavu na konstantní teplotě, částice se pohybují s maxwellovským rozdělením rychlostí. Spočítáme-li průměrnou kinetickou energii všech částic, bude podle ekvipartičního teorému úměrná teplotě a proto konstantní. Pokud ale tentýž postup namodelujeme pomocí metody molekulární dynamiky, dostaneme závislost znázorněnou na obr. 5.3 – teplota souboru částic bude postupně narůstat. Zdůvodnění tohoto jevu spočívá v tom, že jsme původně spojitou časovou osu diskretizovali, čehož důsledkem bude nahrazení původní trajektorie částice ve tvaru hladké křivky trajektorií tvořenou soustavou úseček. Tím může dojít k tomu, že se při modelování změní proti realitě vzájemná poloha částic, vzroste nepřirozeně jejich interakce a tím se může zvýšit i jejich kinetická energie. To je zřejmé též z obr. 5.3, kde pro menší časový krok Δt chyba poklesne. Tato cesta nápravy však k cíli nevede, neboť pro úplné odstranění tohoto jevu by časový krok musel klesnout na tak malou hodnotu, že doba výpočtu by rostla k nekonečnu a zaokrouhlovací chyba by vše překryla. Byly navrženy i jiné postupy k minimalizaci tohoto jevu, žádný však nebyl zcela úspěšný. Nezbude proto nic jiného, než si být existence nefyzikálního ohřevu vědomi a snažit se pracovat s takovou kombinací parametrů modelování, Δt a t_{max}, aby v průběhu výpočtu zůstala změna energie soustavy v rozumných mezích.



Obrázek 5.3: Zvyšování střední energie souboru částic v průběhu modelování metodou molekulární dynamiky. Velikosti časových kroků $\Delta t_1 > \Delta t_2$.

5.3 Metoda P-I-C a další postupy urychlující výpočet

5.3.1 Formulace problému

Metoda molekulární dynamiky dovoluje sice získat detailní informace o studované soustavě, v některých případech však je velice neefektivní. Pokud budeme analyzovat algoritmy na řešení pohybových rovnic (5.1), (5.2) a (5.3) z hlediska časové náročnosti, zjistíme, že převážnou část doby program stráví při výpočtu síly $\vec{F}_i$. Pokud chceme program urychlit, musíme proto pozornost věnovat této části programu.

Sílu působící na i-tou částici počítáme podle vztahu

$$\vec{F}_i = \vec{F}_i^{ext} + \sum_{\substack{j=1 \\ j \neq i}}^{N} \vec{f}_{ij} \,, \qquad i = 1, \ldots, N \,, \tag{5.4}$$

kde $\vec{F}_i^{ext}$ je síla působící na částici z externího zdroje a druhý člen rovnice představuje vzájemné silové působení souboru částic. V případě elektrostatickém můžeme externí sílu vyjádřit pomocí intenzity externího elektrostatického pole (např. v laboratorním nízkoteplotním plazmatu se bude jednat o elektrické pole mezi katodou a anodou)

$$\vec{F}_i^{ext} = q_i \, \vec{E}^{ext}(\vec{r}_i) \,. \tag{5.5}$$

Zde $\vec{E}^{ext}(\vec{r}_i)$ je intenzita vnějšího pole v místě i-té částice $\vec{r}_i$ a q_i je náboj této částice. Jelikož tato intenzita pole závisí jen na poloze i-té částice, bude první člen v soustavě rovnic pro výpočet celkového silového působení (5.4) lineárně záviset na počtu částic N, $O(N)$.

O efektivitě výpočtu proto rozhoduje druhý člen rovnice (5.4). Zde musíme rozlišit případ dalekodosahových a krátkodosahových sil. V případě krátkodosahových sil se ve vzájemném silovém působení projeví pouze částice z nejbližšího okolí sledované částice, proto v součtu $\sum \vec{f}_{ij}$ můžeme zvolit horní mez konstantní, nezávislou na N (a mívající hodnotu 20-30). V tomto případě též druhý člen rovnice (5.4) závisí na počtu částic lineárně a proto i celý výpočet síly má závislost typu $O(N)$. Naproti tomu pro dalekodosahové síly musíme vztah pro výpočet silového působení na i-tou částici vzít ve formě uvedené v rovnici (5.4) a důsledkem bude kvadratická závislost na počtu částic v souboru $O(N^2)$.

V klasické definici metody molekulární dynamiky vystupují síly krátkodosahové a pro ně stačí řešit pohybové rovnice algoritmy (5.2) a (5.3), do kterých se dosadí silové působení z rovnice (5.4) (s konstantní horní mezí).

Pro problémy z oblasti astronomie, fyziky plazmatu a podobných oblastí, je nutno definici metody molekulární dynamiky rozšířit a zahrnout do ní efektivnější metody pro výpočet síly.

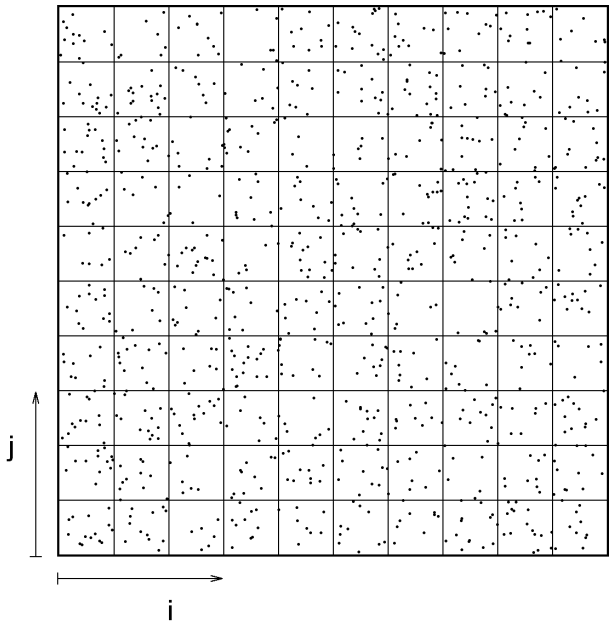
5.3.2 Metoda P-I-C

Problém přibližného počítání silového působení je již dávno ve fyzice řešen v tzv. *problému mnoha těles* (např. [15]). Byla vyvinuta řada efektivních algoritmů, které vedou na výpočet síly lepší než $O(N^2)$.

Základní myšlenka většiny těchto algoritmů spočívá v tom, že vzájemné silové působení ve vztahu pro sílu (5.4) nahradíme ekvivalentem vztahu (5.5). K tomu účelu musíme najít intenzitu lokálního elektrického nebo gravitačního pole, $\vec{E}^{lok}$, a pak již můžeme použít jednodušší vztah pro výsledné silové působení

$$\vec{F}_i(\vec{r}_i) = \vec{F}_i^{ext}(\vec{r}_i) + \vec{F}_i^{lok}(\vec{r}_i)\,, \qquad i = 1, \ldots, N\,.$$

Sám tento vztah je v závislosti na počtu částic lineární, přičítá k tomu však ještě doba potřebná k určení intenzity lokálního pole, takže jako výsledek efektivita různých algoritmů na výpočet síly leží v rozmezí $O(N \cdot \log N)$ až dokonce $O(N)$.



Obrázek 5.4: Pracovní oblast pro metodu P-I-C.

Nejjednodušší a ve většině případů postačující je metoda P-I-C, neboli „Particle-In-Cell“. V této metodě navíc k obvyklé diskretizaci časové osy (s

krokem Δt), jež je standardním postupem v metodě molekulární dynamiky, provedeme ještě diskretizaci prostoru. V závislosti na tvaru pracovní oblasti tuto oblast rozdělíme na soustavu menších buněk - viz obr. 5.4, kde jsme použili čtvercové buňky s hranou o velikosti Δx. Původní soubor N částic se nám tak rozdělí mezi jednotlivé buňky (odtud pochází název metody).

Další postup výpočtu intenzity lokálního pole bude následující (výklad povedeme pro pole elektrostatické, obdobně bude probíhat i pro pole gravitační):

1. Elektrický náboj všech částic v buňce složíme do jedné výsledné částice ve středu buňky q_{ij}, kde (ij) jsou souřadnice příslušné buňky – konkrétní postup bude popsán dále.

2. Vydělením objemem buňky přejdeme od nábojů q_{ij} k prostorové hustotě náboje ϱ_{ij}.

3. Vyřešíme Poissonovu rovnici

$$\Delta U = -\frac{\varrho}{\epsilon_0}$$

   a dostaneme hodnoty elektrického potenciálu v každé buňce U_{ij}.

4. Diferenčními schématy typu

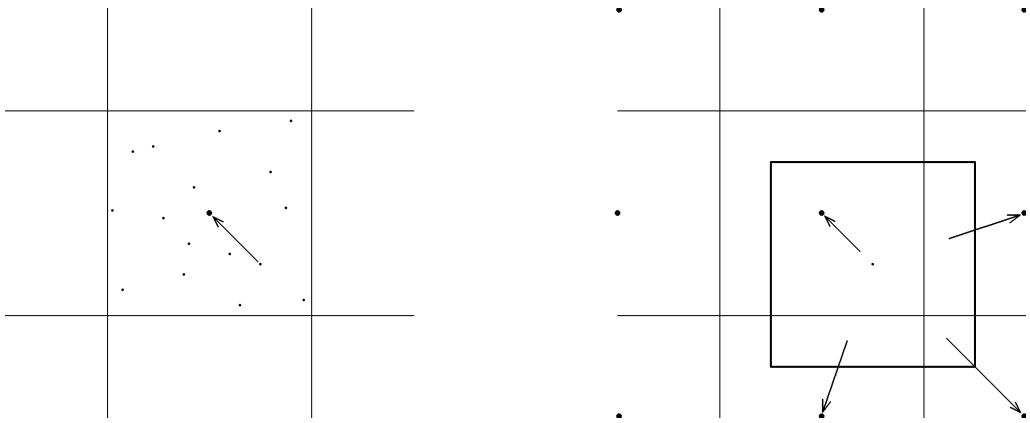
$$(E_{ij})_x = \frac{U_{i+1,j} - U_{i-1,j}}{2\,\Delta x}$$

   přejdeme od elektrostatického potenciálu k hledané intenzitě elektrického pole $\vec{E}_{ij}^{lok}$ v jednotlivých buňkách.

Nyní se vraťme ke kroku 1. Celkový elektrický náboj v buňce q_{ij} můžeme získat dvěma postupy – viz obr. 5.5:

– Algoritmus NGP
  Metoda NGP („Nearest Grid Point“) znamená prosté sečtení všech nábojů v buňce (s ohledem na jejich znaménko) a přenesení výsledného náboje do středu buňky.

– Algoritmus CIC
  V metodě CIC („Cloud In Cell“) předpokládáme, že náboj není bodový, ale má tvar oblaku (anglicky „Cloud“ a odtud pochází název tohoto algoritmu). Oblak obvykle zasahuje do více buněk a přispívá proto k několika výsledným nábojům ve středu buněk úměrně svému objemu v příslušné buňce. V metodě CIC se pro jednoduchost předpokládá, že nábojový mrak má konstantní hustotu a tvar stejný jako buňka, tj. je obvykle krychlový nebo čtvercový.



Obrázek 5.5: Princip skládání náboje metodou NGP (vlevo) a CIC (vpravo).

Metoda NGP vede na rozložení náboje s podstatně většími fluktuacemi než dostaneme při použití metody CIC, což se v závěru promítne i do většího rozptylu hodnot intenzit lokálního elektrického pole. Proto pohybové rovnice můžeme při použití metody CIC integrovat s větším časovým krokem, což více než vykompenzuje poněkud pomalejší skládání náboje ve srovnání s metodou NGP. Metoda CIC v podstatě představuje lineární interpolaci náboje. V současné době se někdy používají i silnější interpolační metody – kvadratická, pomocí splinů, apod., obvykle však stačí pracovat s metodou CIC – přesněji s metodou P-I-C v modifikaci CIC.

Metoda P-I-C znamená z hlediska výpočtu trajektorií souboru vzájemně interagujících částic pouze dílčí algoritmus pro nalezení silového působení na i-tou částici, $\vec{F}_i$. Tuto sílu nyní musíme dosadit do pohybových rovnic – např. do Verletova algoritmu (5.2) – a provést integrování celé soustavy rovnic v požadovaném časovém intervalu $\langle t_0, t_{max} \rangle$.

Závislost doby výpočtu silového působení metodou P-I-C je ve srovnání s původním vektorovým součtem všech dílčích sil (tj. principem superpozice) mnohem efektivnější. Jak jsme již uvedli výše, tato doba se skládá ze dvou složek – z výpočtu síly působící na částici v lokálním poli, což vede na lineární závislost na počtu částic N, a z doby potřebné na určení tohoto pole. I tento člen závisí na N, neboť podle počtu částic volíme hustotu prostorové sítě. Pokud bude krok Δx příliš velký, bude spočítáno prostorové rozložení lokálního pole příliš hrubě, naopak pokud zvolíme síť příliš jemnou ve srovnání s počtem částic, prudce narostou fluktuace v intenzitách pole mezi jednotlivými buňkami sítě. Empirický odhad proto radí volit velikost kroku Δx tak, aby v jedné buňce bylo řádově 10^1 částic. Pokud tyto odhady složíme dohromady, dostaneme pro efektivitu metody P-I-C závislost typu $O(N \cdot \log N)$.

Ještě je nutno uvést fyzikální důsledek toho, že jsme urychlili výpočet síly v metodě molekulární dynamiky. Na jedné straně jsme pro dalekodosahové síly dostali podstatně silnější metodu, která nám umožní řešit i problémy původní metodou neřešitelné, platíme však za to dalším snížením přesnosti výpočtu. Podobně jako v původní metodě, kde po zavedení diskretizace času není možno analyzovat příliš rychlé procesy, nyní navíc nelze studovat procesy, které probíhají v prostorové škále menší než Δx.

5.3.3 Další postupy

Modifikace P-I-C přináší proti původní metodě molekulární dynamiky efektivnější výpočetní prostředek, ten se ale projeví jen pro určité počty částic N. Je nutno si uvědomit, že postup vedoucí na určení lokálního elektrického nebo gravitačního pole je časově náročný, přičemž nejnáročnější je řešení Poissonovy rovnice. Pokud srovnáme závislost typu $O(N^2)$ pro původní metodu a závislost $O(N \cdot \log N)$ pro její modifikaci, tak vedle vlastní funkční závislosti se též projeví multiplikativní konstanta ve vztahu pro efektivitu výpočtu postupem P-I-C, a tu můžeme odhadnout pouze přibližně. Její hodnota bude záviset na tom, jakým postupem budeme řešit Poissovovu rovnici, zda budeme problém formulovat v jedné, dvou nebo třech dimenzích, atd. Velmi přibližně můžeme odhadnout, že hranice výhodnosti použití jednotlivých postupů leží v okolí $1 \cdot 10^3$ částic – pro menší počet částic je výhodnější pracovat s původní metodou molekulární dynamiky zatímco pro větší N začne převažovat výhodnost modifikace P-I-C. Při praktickém použití se celé rozhodování zjednodušuje. Obvykle máme buď problémy obsahující pouze několik desítek interagujících částic (studium interakce atomů v kapalinách a pevných látkách nebo pohybu těles ve Sluneční soustavě) a pak je nutno použít metodu molekulární dynamiky v klasické formulaci, a nebo máme 10^5 až 10^8 částic (výpočty ve fyzice plazmatu, kinetické teorii plynů, stelární astronomii nebo kosmologii) a pak původní metodu nelze vůbec použít, neboť rozdíl funkčních závislosti mezi N^2 a $N \cdot \log N$ je obrovský a tvoří několik řádů.

Pro skutečně velké problémy, kde počet interagujících částic výrazně překračuje hodnotu 10^6, přestává být použitelná i metoda P-I-C a je nutno hledat ještě silnější algoritmy na výpočet vzájemného silového působení těchto částic.

K tomuto účelu bylo navrženo a je prakticky používáno několik desítek dalších algoritmů nebo jejich modifikací. Většina z nich je založena též na prostorovém členění pracovní oblasti podobně jako na obr. 5.4, ale složitějším způsobem. Slabinou metody P-I-C je, že jsme náboje/tělesa posunuli z jejich správných poloh do středů buněk a

důsledkem toho bude, že spočítaná hodnota intenzity lokálního pole bude odpovídat této změněné konfiguraci místo původnímu rozložení částic. Pro částice vzdálené od naší i-té částice, pro níž počítáme sílu, je tato chyba zanedbatelná, zatímco pro bližší buňky a částice v nich se zvětšuje. Byly proto navrženy různé hierarchické metody, které pracují se sítí, jež se postupně zjemňuje, aby kompenzovala relativní chybu v změněných polohách blízkých částic. Nevýhodou tohoto postupu je, že takové zjemňování sítě se musí provádět individuálně pro každou částici, na níž počítáme silové působení. Další metody proto provádějí, podstatně složitějším způsobem, zjemňování použitelné pro všechny částice. V převážné většině těchto metod po vytvoření více nebo méně komplikované sítě postupujeme podobně jako v metodě P-I-C, tj. určíme hodnotu lokálního pole. Existují však i postupy, které z náboje v síti počítají sílu přímo – např. pomocí multipólového rozvoje a nebo rovnou pomocí principu superpozice.

Tyto silnější postupy mohou opět vést na efektivitu typu $O(N^2)$ a $O(N \cdot \log N)$, ale s výrazně menší multiplikativní konstantou. Nejlepší z nich však dosahují až limitně nejdokonalejší závislosti $O(N)$. Na rozdíl od metody P-I-C však jsou tyto postupy programátorsky výrazně složitější, některé z nich představují až roky práce na rozdíl od několika desítek hodin algoritmu P-I-C, takže je vhodné je využívat pouze jako předem připravené knihovní podprogramy. Jako příklad složitějších algoritmů si můžeme uvést např. metodu P^3M („Particle-Particle-Particle-Mesh“), Ewaldovu sumaci, Barnesův-Hutův algoritmus nebo rychlou multipólovou metodu FMM [16], [17], [18], [19].

5.4 Shrnutí

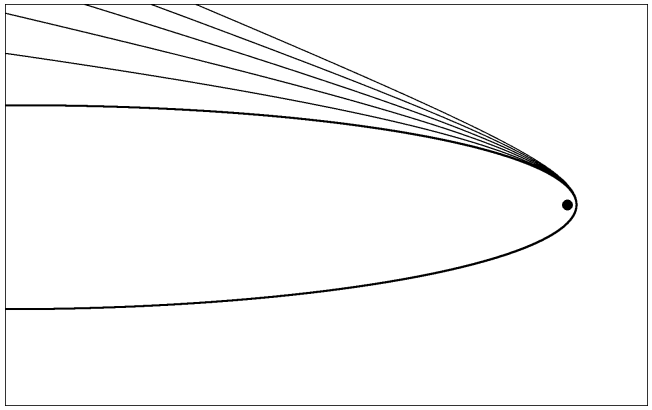
Z oblasti deterministického částicového modelování lze považovat za nejpodstatnější tyto okruhy znalostí:

- **Princip metody molekulární dynamiky, její přednosti a slabiny**
- **Volba pracovní oblasti, okrajových podmínek a určování silového působení**
- **Základní algoritmy pro řešení pohybových rovnic, srovnání oblastí použití**
- **Metoda P-I-C včetně algoritmů NGP a CIC.**

5.5 Problémy k řešení

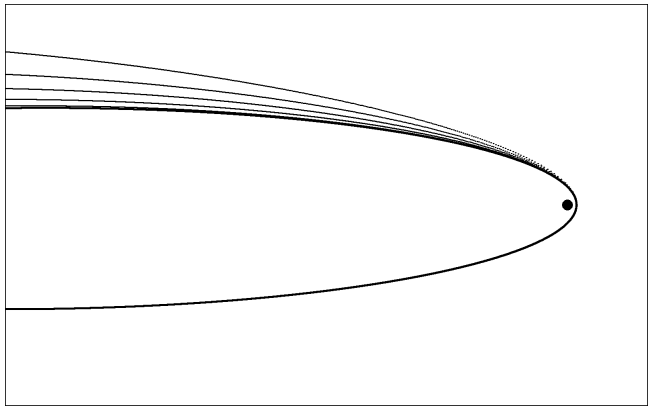
Problémy určené k procvičování látky popsané v této kapitole buď formou samostudia nebo v rámci organizované výuky:

1. Vypočtěte a nakreslete s použitím Eulerova algoritmu trajektorii komety obíhající Slunce. Při výpočtu měňte lineárně časový krok a sledujte chybu řešení.



Obrázek 5.6: Trajektorie pro krok $\Delta t = (1 \div 5) \cdot \Delta t_0$ a přesná trajektorie.

2. Vypočtěte a nakreslete trajektorii komety obíhající Slunce s použitím algoritmu druhého řádu. Při výpočtu měňte lineárně časový krok a sledujte chybu řešení.



Obrázek 5.7: Trajektorie pro krok $\Delta t = (1 \div 5) \cdot \Delta t_0$ a přesná trajektorie. Základní krok Δt_0 je 10x větší než v předchozím problému.

3. Najděte a otestujte rovnovážné body v soustavě Slunce-Jupiter.
   Podle Lagrangeova řešení problému tří těles [15], v soustavě dvou hmotných těles obíhajících kolem společného těžiště existuje pět rovnovážných bodů. Tyto body mají takovou vlastnost, že když do nich umístíme těleso zanedbatelné hmotnosti a udělíme mi patřičnou rychlost, tak zůstane vůči dvěma základním tělesům v klidu. Tři z těch bodů se vyznačují rovnováhou labilní, což nemá praktický význam. Zbylé dva Lagrangeovy body jsou však body rovnováhy stabilní, přičemž jejich stabilita je tím větší, čím větší jsou hmotnosti základních dvou těles (ve Sluneční soustavě proto nejstabilnější Lagrangeovy body jsou v soustavě Slunce-Jupiter.)

4. Namodelujte chování trojhvězdy, kde všechny tři hvězdy mají stejnou hmotnost. Pracujte s různými počátečními polohami a rychlostmi jednotlivých hvězd.
   Z praktických důvodů volte polohy a rychlosti hvězd tak, aby těžiště soustavy bylo ve středu obrazovky a zůstávalo v klidu.

5. Namodelujte planetární soustavu dvouhvězdy.
   Řešte dva možné případy:
   – hvězdy dosti vzdálené a každá z nich má vlastní planetární soustavu,
   – hvězdy blíže u sebe a existují planety příslušející jak k jednotlivým hvězdám tak planety obíhající obě hvězdy.
   Otestujte, zda existuje stabilní řešení. Pokud takové řešení naleznete, sledujte jak se stabilita bude narušovat numericky se zvětšováním časového kroku Δt.

6. Namodelujte v reálném měřítku soustavu Slunce-Země.
   Použijte následující konstanty:
   $M_S = 1,990 \cdot 10^{30}$ kg – hmotnost Slunce
   $M_Z = 5,976 \cdot 10^{24}$ kg – hmotnost Země
   $d_{SZ} = 1,496 \cdot 10^{11}$ m – vzdálenost Slunce-Země.
   Otestujte numerickou stabilitu této soustavy v závislosti na velikosti časového kroku Δt.

7. Do modelu z předchozího příkladu přidejte „Antizemi“ a otestujte, zda soustava zůstane stabilní.
   Antizemě je fiktivní planeta popsaná v literatuře, jež je velikosti Země a pohybuje se na oběžné dráze Země, ale je stále schována za Sluncem (na oběžné dráze je posunuta o 180°).
   Pokuste se stabilitu dále narušit přidáním dalších planet do soustavy.

Kapitola 6

SPOJITÉ MODELOVÁNÍ A HYBRIDNÍ POSTUPY

V třetí kapitole jsme uvedli přehled jednotlivých technik počítačového modelování ve fyzice. Při postupech částicového modelování, jejichž popisu jsme se věnovali v předchozích kapitolách, popisujeme studovaný jev na mikroskopické úrovni. Naproti tomu spojité modelování studuje fyzikální proces na úrovni makroskopické. Kombinace více postupů s cílem využít jejich přednosti a aspoň částečně potlačit jejich nevýhody se pak nazývá modelování hybridní. Těmto technikám bude věnována poslední kapitola prvního dílu studijního textu s tím, že jak spojité tak i hybridní modelování překračuje bakalářskou a dokonce i magisterskou úroveň a proto tato kapitola bude sloužit pouze jako úvod do studované problematiky.

6.1 Spojité modelování

Pokud budeme chtít studovat fyzikální jev na makroskopické úrovni, budeme pracovat s makroskopickými veličinami typu teplota, tlak, koncentrace, rychlost proudění, elektrická, magnetická a gravitační pole, teplota, atd. Mezi těmito veličinami platí vztahy obvykle popisované rovnicemi matematické fyziky, jedná se o zákony zachování energie, hybnosti, hmoty, náboje, apod. Většina těchto rovnic je již v teoretické fyzice známa včetně základních metod jejich řešení – rovnice spojitého modelování jsou nejčastěji parciálními diferenciálními rovnicemi a k jejich řešení můžeme použít postupy numerické matematiky včetně v tomto studijním textu uvedené stochastické metody řešení.

Popis studovaného jevu na úrovni spojité většinou odpovídá úrovni experimentální fyziky, neboť i výsledky měření poskytují údaje o vztazích mezi

podobnými veličinami – teplotní závislosti elektrického proudu, tlaková závislost koncentrace částic, atd. V této oblasti je proto vazba mezi výsledky modelování a měření dosti jednoduchá. Naproti tomu při částicovém modelování získáváme obvykle informace z hlediska experimentátora přímo nepoužitelné. Pokud známe polohy a rychlosti všech mikroskopických částic v jednotce objemu, pro srovnání s experimentem je nejprve musíme středováním převést na koncentrace, tlak, teplotu, ... a pak teprve porovnávat a diskutovat. V procesu tohoto středování se část modelováním získané informace potlačí.

Při srovnání technik částicového a spojitého modelování můžeme proto shrnout výhody a nevýhody obou přístupů:

– Částicové modelování
  Pozitivní vlastnost: Přináší detailní informace o studované soustavě na mikroskopické úrovni, proto získané výsledky mají mnohem větší vypovídací hodnotu než v případě modelování spojitého.
  Negativní vlastnosti: Programy (zejména pro velký počet částic) jsou výpočetně velmi náročné a musí se proto hledat dosti sofistikované algoritmy pro urychlování výpočtu. Obvykle neexistuje přímá vazba na experiment.

– Spojité modelování
  Pozitivní vlastnosti: Programy jsou obvykle výpočetně velmi málo náročné – typicky minuty až desítky minut strojového času proti dnům až měsícům u modelování částicového. Srovnání výsledků modelování s experimentem bývá jednoduché.
  Negativní vlastnosti: Použité rovnice matematické fyziky jsou velmi obecné a i přes jejich konkretizaci volbou počátečních a okrajových podmínek jsou získávané výsledky dosti obecné a nepřesné. Rovnice spojitého modelování, i když jich je v modelu výrazně méně (typicky $2 \div 10$ proti 10^4 až 10^8 u modelování částicového), jsou podstatně složitější a nalezení algoritmu řešení včetně naprogramování bývá dosti obtížné a časově náročné.

Jako příklady spojitého modelování si je možno uvést většinu postupů uváděných v teoretické fyzice, např. řešení soustavy Maxwellových rovnic v teorii elektromagnetického pole.

> Jeden z problémů, který bývá řešen technikou spojitého modelování, je studium chemické kinetiky. V plazmochemii dochází k tomu, že ve výboji se složitější sloučeniny z plazmatu rozkládají a v následujícím řetězu reakcí vytvářejí sloučeniny nové. Příkladem takových procesů může být vytváření diamantových vrstev výbojem v uhlovodících nebo vznik a zánik ozónu v kyslíkovém výboji pod vlivem dalších příměsí jako je metan nebo freony.

Označme symboly n_i koncentrace jednotlivých složek ve výboji. Soustavu rovnic, jejich řešením dostaneme popis chemické kinetiky, vytvoříme na základě rovnic kontinuity. Rovnice kontinuity pro jednotlivé složky plazmatu představují zákony zachování počtu příslušných typů částic. Tyto rovnice mají pro i-tou složku obecný tvar

$$\frac{\partial n_i}{\partial t} = A_i + B_i + difuze + drift\,.$$

Zde výrazy A_i a B_i představují změnu příslušné složky plazmatu n_i způsobenou chemickými reakcemi v jednotkovém objemu plazmatu a symbolické členy *difuze* a *drift* představují změnu této složky vlivem uvedených prostorových procesů. Koncentrace n_i se budou obecně měnit jako funkce času i prostorových souřadnic, proto bude model vyjádřen v parciálních diferenciálních rovnicích.

Budeme-li řešit stacionární případ, při kterém drift a difuze nebudou přítomny, resp. tyto procesy budou vyváženy takže je v modelu nemusíme uvažovat, dostaneme soustavu m obyčejných diferenciálních rovnic (m je celkový počet složek ve výboji)

$$\frac{\mathrm{d}n_i}{\mathrm{d}t} = A_i + B_i\,, \qquad i = 1, \ldots, m\,,$$

kde člen $A_i(n_1, \ldots, n_m, T_e, T_g)$ popisuje rychlost vzniku částic i-tého typu a $B_i(n_1, \ldots, n_m, T_e, T_g)$ rychlost jejich zániku. Tyto veličiny závisejí na koncentraci jednotlivých složek ve formě výrazů typu $k_1 \cdot n_1$, $k_2 \cdot n_1 \cdot n_2$ a $k_3 \cdot n_1 \cdot n_2 \cdot n_3$, kde rychlostní koeficienty jednotlivých reakcí mohou být funkcemi teploty elektronů T_e nebo teploty plynu, resp. iontů, T_g.

Vybudujeme-li model chemické kinetiky na základě souboru chemických rovnic včetně počátečních hodnot jednotlivých složek plazmatu, dokážeme sestavit soustavu obyčejných diferenciálních rovnic typu $\mathrm{d}n_i/\mathrm{d}t = \cdots$. Tyto rovnice jsou nelineární, neboť obsahují na pravé straně členy typu $k_2\, n_1\, n_2$, atd. Typický počet rovnic v takovém souboru bývá několik desítek, známé však jsou i extrémní případy s více než 1 600 rovnicemi.

Pokud budeme chtít takové soustavy rovnic řešit, nestačí k tomu obvyklé znalosti z numerické matematiky, neboť metody typu Rungeho-Kutty nebo prediktor-korektor jsou použitelné maximálně do 10-20 rovnic. Z matematického hlediska spočívá problém v tom, že vytvořená soustava rovnic má velkou tuhost (jedná se o tzv. „stiff“ soustavu). Proto je nutné použít specializovaný software [20].

6.2 Hybridní modelování

Základní myšlenkou hybridního modelování je využít pro řešení problému kombinaci několika principů s cílem potlačit nevýhody obou dílčích technik a využít jejich přednosti. V rámci doposud probíraných metod počítačového modelování ve fyzice můžeme mluvit o dvou úrovních hybridního modelování:

- Částicová úroveň hybridního modelování
  Při tomto postupu kombinujeme pouze dva přístupy částicového modelování, tj. modelování deterministické (metodu molekulární dynamiky) a modelování stochastické (metodu Monte Carlo).

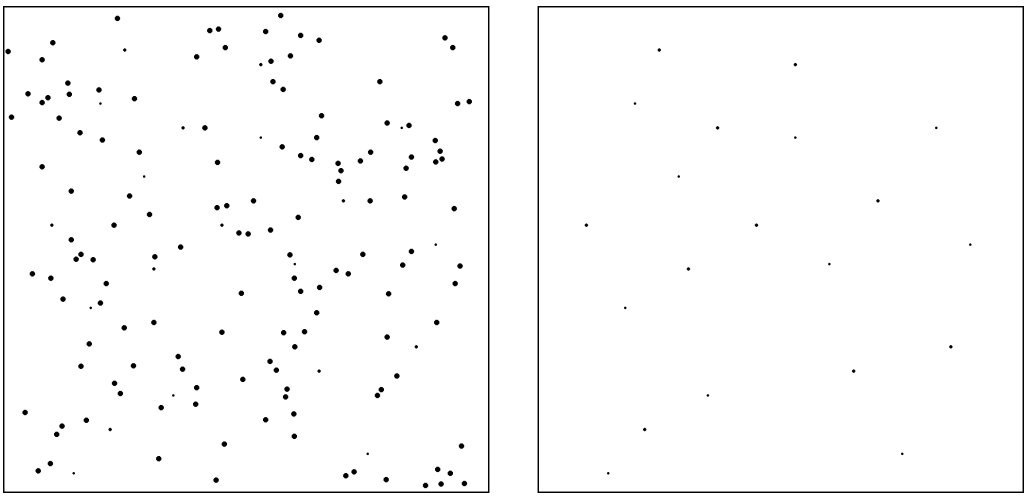
- Úplné hybridní modelování
  Při něm kombinujeme modelování spojité s modelováním částicovým. Z modelování spojitého se snažíme převzít rychlost výpočtu a z modelování částicového přesnost.

Výsledkem kombinování dvou a více technik bývá kompromis – výsledek obvykle nedosahuje špičkových parametrů (v rychlosti, v přesnosti) žádného z obou původních postupů, nemá však ani jejich výrazné slabiny. Za toto zlepšení modelování však ale obvykle platíme složitostí přípravy modelu a výsledného programu, která bývá nejméně stejná jako součet obou technik, často však jej výrazně překračuje. Proto též hybridní modelování jako nejsložitější proces obvykle patří až do doktorské úrovně studia – to platí především pro úplné hybridní modelování.

Závěrem si uveďme příklad hybridního modelování na částicové úrovni. Již jsme si řekli, že určitá část problémů deterministického modelování se řeší na skutečné úrovni mikrosvěta, kdy studovanými částicemi („molekulami“) jsou atomy, elektrony a ionty. V tomto případě nemáme žádnou možnost získat počáteční podmínky (souřadnice a rychlosti) přímým pozorováním a je nutno je rozehrát stochasticky na základě známých rozdělovacích funkcí, získaných buď z teoretických úvah nebo i z měření. Již toto ve skutečnosti představuje hybridní přístup modelování, i když to autoři často neberou v úvahu.

Skutečně částicové hybridní modelování je základní technikou modelování v nízkoteplotním plazmatu. V grafické formě je tato technika znázorněna na obr. 6.1.

Nízkoteplotní plazma představuje směs neutrálních a nabitých částic, přičemž celkový počet nabitých částic ve směsi představuje jen velmi malou část – typicky $10^{-4} \div 10^{-6}$ (tzv. stupeň ionizace plazmatu). O chování plazmatu rozhoduje jeho nabitá složka a proto právě tu musíme modelovat. Zcela realistický částicový model plazmatu můžeme vybudovat jen pomocí aparátu metody molekulární dynamiky (až na výše zmíněné nastavení počátečních poloh



Obrázek 6.1: Částicový model plazmatu - technika molekulární dynamiky (vlevo) a technika hybridní (vpravo).

a rychlostí částic), tento výpočet bude ale extrémně pomalý, resp. nebude při použití současné výpočetní techniky vůbec možný. I kdybychom se spokojili jen s velmi omezenou přesností, musíme pracovat nejméně s $N_1 = 1 \cdot 10^3$ nabitými částicemi, a pro stupeň ionizace rovný $1 \cdot 10^{-5}$ budeme v modelu mít nejméně $N = 1 \cdot 10^8$ částic. Symbolicky je tento případ znázorněn v levé části obr. 6.1. Čistě molekulárně-dynamický model plazmatu představuje úlohu s N částicemi, které se pohybují ve vakuu pod vlivem vnějších polí a vzájemné interakce.

Tento přístup je mimořádně pracný a přitom zavedení neutrálních částic do modelu je téměř zbytečné. Jejich jedinou úlohou je interagovat s nabitými částicemi a měnit jejich trajektorie, případně i energie. Hybridní přístup proto pracuje pouze s nabitými částicemi a neutrální částice přesune do vlastností „prostředí“. Pohyb nabitých částic budeme studovat místo ve vakuu v látkovém prostředí, jehož vlastnosti budou odpovídat interakcím s neutrálními částicemi. Toto je typická technika transportního problému metody Monte Carlo. Interakce nabitých částic s neutrálními popíšeme pomocí středních volných drah pro tyto interakce. Model pak bude tvořen dvěma částmi:

– Molekulárně-dynamická část:
  Bude se jednat o řídící program, v kterém budeme řešit pohybové rovnice typu (5.2), (5.3) nebo (5.1). Ve členu silového působení budeme pouze uvažovat interakci s vnějšími poli a vzájemné interakce nabitých částic.

– Monte Carlo část:
Pro každou nabitou částici rozehrajeme náhodnou volnou dráhu s charakteristikami danými interakcí s neutrálními částicemi. Pak bude probíhat výpočet v rámci hlavního programu, tj. budeme počítat trajektorie částic na základě pohybových rovnic. Současně však budeme testovat, zda již délka trajektorie od poslední interakce není rovna rozehrané náhodné volné dráze. Pokud tento případ nastane, přejdeme do podprogramu Monte Carlo, rozhodneme o typu interakce mezi nabitou a neutrální částicí, odpovídajícím způsobem ošetříme energii a směr pohybu částice, rozehrajeme novou náhodnou volnou dráhu a vrátíme se zpět do hlavního molekulárně-dynamického programu.

Tento přístup znamená velmi značné urychlení výpočtu. Celkový počet částic v modelu klesne, neboť bude obsahovat pouze nabité částice (tj. N_1). Výpočet se sice poněkud zpomalí zavedením Monte Carlo složky do modelu, zefektivnění výpočtu poklesem počtu částic však bude velmi výrazně převažovat. Toto je symbolicky znázorněno v pravé části obr. 6.1.

6.3 Shrnutí

Tato kapitola má pouze přehledový charakter, proto za podstatné lze považovat pouze:

- **Základní myšlenky spojitého modelování**
- **Základní myšlenky hybridního modelování.**

Kapitola 7

ZÁVĚR

Tato první část studijního textu byla věnována dvěma základním okruhům otázek:

- Počítačová fyzika:
  V této prvním okruhu tvořeném kapitolami 1 a 2 byl zaveden pojem počítačová fyzika a byly stručně uvedeny hardwarové a softwarové základy této vědní disciplíny.
- Počítačové modelování:
  Druhý okruh tvořící náplň kapitol 3 až 6 byl věnován nejdůležitější partii počítačové fyziky – počítačovému modelování. Relativně podrobně jsou probrány oblasti částicového modelování, metoda Monte Carlo a metoda molekulární dynamiky. Tyto znalosti by měly být postačující k tomu, aby byl čtenář schopen prakticky řešit fyzikální problémy těmito technikami. V poslední šesté kapitole jsou pouze pro informaci uvedeny další postupy počítačového modelování, modelování spojité a hybridní.

V druhé části studijního textu budou relativně podrobně popsány další důležité partie počítačové fyziky – počítačová grafika včetně vizualizace, zpracování obrazu a integrální transformace. V závěru tohoto dílu budou opět na informativní úrovni popsány partie počítačové fyziky buď méně důležité a nebo se teprve rozvíjející.

Literatura

[1] Kernighan B.W., Ritchie D.M.: *The C Programming Language*, Prentice Hall, USA 1978.

[2] Wirth N.: *Algorithms + Data Structures = Programs.*
Přepracované vydání – Wirth N.: *Algorithms and Data Structures*, Prentice Hall, USA 1986.

[3] Metropolis N., Ulam S.: *The Monte Carlo Method*, J. Amer. Statist. Assoc. **44** (1949), No 247, 335.

[4] Hall A.: *On the Experimental Determination of* π, Messeng. Math. **2** (1873), 113.

[5] Buffon G.L.L.: *Essai d'arithmétique morale*, Suppl. à l'Histoire Naturelle, Vol. 4, Imp. Royale, Paris 1777.

[6] Anděl J.: *Statistické metody*, Matfyzpress, Praha 1993.

[7] Olehla M., Věchet V., Olehla J.: *Řešení úloh matematické statistiky ve FORTRANU*, NADAS, Praha 1982.

[8] Hrach R.: *Numerické metody ve fyzikální elektronice I*, skripta MFF UK, SPN, Praha 1981.

[9] Moore W.J.: *Fyzikální chemie*, SNTL, Praha 1979.

[10] Ripley B.D.: *Stochastic Simulation*, John Wiley and Sons, New York 1987.

[11] Müller M.E.: Ann. Math. Stat. **29** (1958), 610.

[12] Skullerud H.R.: *Null collision method*, Techn. Report EIP 72-1, Norwegian Inst. of Technology, Trondheim 1972.

[13] Nanbu K.: Jpn. J. Appl. Phys. **33** (1994), 4752.

[14] Eckertová L., Frei V., Hájek Z., Hrach R., Malát V., Tomková E.: *Fyzikální elektronika pevných látek*, Karolinum, Praha 1992.

[15] Vanýsek V.: *Základy astronomie a astrofyziky*, Academia, Praha 1980.

[16] Hockney R.W., Eastwood J.W.: *Computer Simulation Using Particles*, Adam Hilger, Bristol 1988.

[17] Greengard L., Rokhlin V.: J. Comput. Phys. **73** (1987), 325.

[18] Barnes J., Hut P.: Nature **324** (1986), 446.

[19] Petersen H.G.: J. Chem. Phys. **103** (1995), 3668.

[20] Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P.: *Numerical Recipes in FORTRAN. The Art of Scientific Computing. Second Edition*, Cambridge University Press, Cambridge 1992.