Osnove numeričnega reševanja fizikalnih problemov

Bojan Golli, Pedagoška fakulteta, Univerza v Ljubljani

Kazalo

1. Nekatere metode za reševanje navadnih diferencialnih enačb

1.1 Diskretizacija

Fizikalne zakone običajno zapišemo v obliki diferencialne enačbe. Pri preprostih problemih znamo rešitev poiskati analitično; že pri nekoliko bolj realističnih primerih pa to ni več mogoče in tedaj nam ostane le numerično reševanje.

Najprej na kratko predstavimo postopek numeričnega reševanja diferencialnih enačb in v nadaljevanju nekatere najpreprostejše metode, prilagojene različnim zvrstem enačb. Za zgled vzemimo kar enačbo za nihanje telesa z maso m na vijačni vzmeti s konstanto k:

$$ma = -ks \qquad \text{ali} \qquad m\frac{d^2s}{dt^2} = -ks\,. \tag{1}$$

K enačbi sodi še začetni pogoj; običajno je podan začetni odmik s_0 in začetna hitrost v_0.

1. Na začetku prepišimo enačbo v *brezdimenzijsko obliko*. Pri našem zgledu bi utegnila biti primerna izbira $y = s/s_0$, če le $s_0 \neq 0$, in $x = \omega t$, $\omega = \sqrt{k/m}$.

2. Rešitev iščemo v obliki tabele, ki podaja vrednosti odvisne spremenljivke (spremenljivk) v izbranih točkah: $s_i = s(t_i)$. Točke t_i dobimo tako, da interval $[t_0, t_N]$ razdelimo na N podintervalov. Običajno vzamemo, da so točke porazdeljene ekvidistantno: $t_i = i \cdot h,\ i = 0, \ldots N$, h je korak, $h = (t_N - t_0)/N$, vseh točk pa je seveda $N+1$.

3. Vedno iščemo le *partikularno* rešitev; ta je določena z *robnimi* (*začetnimi*) pogoji. Število pogojev mora biti enako redu enačbe.

4. Velika večina numeričnih metod je namenjena reševanju enačb *prvega reda*. To pravzaprav ni nikakršna omejitev, saj je mogoče poljubno enačbo n-tega reda prepisati v sistem n enačb prvega reda. Pri našem zgledu enačbo (1) prepišemo v sistem enačb:

$$\frac{ds}{dt} = v\,, \qquad \frac{dv}{dt} = -\frac{k}{m}s\,. \tag{2}$$

5. Diferencialno enačbo prepišemo v *diferenčno* enačbo. Postopek ni enoličen in različne sheme vodijo do boljših ali slabših numeričnih algoritmov. Dober algoritem je *točen* in *stabilen*.

V nadaljevanju predstavimo nekaj najbolj pogostih shem. Zglede bomo napisali v programskem jeziku Pascal, ki je v šoli najbolj razširjen programski jezik. Turbo Pascal, ki ga najdemo na večini osebnih računalnikov, tudi omogoča enostavno delo z grafiko, kar nam bo prišlo še posebej prav pri pisanju demonstracijskih programov. Fiziki sicer še vedno najbolj uporabljamo FORTRAN, ki je najprimernejši za pisanje velikih programskih paketov, kjer je odločilna hitrost izvajanja in raba velikih polj. Resnih projektov seveda ne pišemo v BASICu; za pisanje šolskih demonstracijskih programov pa utegne biti zelo priročen.

1.2 Diferencialne enačbe prvega reda (enačbe hoda)

Najbolj preprosta shema, ki prevede diferencialno enačbo v diferenčno, je *Eulerjeva metoda*. Izpeljemo kar iz običajne definicije odvoda, tako da v enačbi

$$\frac{dy}{dx} = f(x,y) \tag{3}$$

odvod v točki x_i nadomestimo z znanim izrazom

$$\frac{y(x_i+h)-y(x_i)}{h} = \frac{y_{i+1}-y_i}{h} = f(x_i,y_i)\ , \tag{4}$$

od koder takoj razberemo uporaben postopek

$$y_{i+1} = y_i + h\cdot f(x_i,y_i)\ , \tag{5}$$

saj začetno vrednost y_0 poznamo iz začetnega pogoja.

Kako je z natačnostjo metode? Če v (4) izraz $y(x_i+h)$ razvijemo po Taylorju do člena s h^2, vidimo $y(x_i+h)-y_{i+1} = \frac{1}{2}y''(x_i)h^2$. Približek (4) za odvod je tem boljši, čim manjši je korak h. Napaka je sorazmerna s h^2; za takšno metodo pravimo, da je *prvega reda*. V splošnem velja, da je pri metodi n-tega reda napaka sorazmerna s h^{n+1}. Pričakujemo torej, da bomo z manjšim korakom dobili natačnejši rezultat. Vendar si premajhnega koraka ne moremo privoščiti. Problem ni le v časovni zahtevnosti, ampka predvsem v zaokrožitvenih napak pri računanju. Pri enačbi prvega reda zato prej ali slej dobimo primes rešitve, ki eksponentno narašča. Stabilnost lahko povečamo, če računamo v dvojni natančnosti, vendar to upočasni hitrost računanja.

Red lahko izboljšamo pri *sredinski metodi*, pri kateri izraz (4) nadomestimo z

$$\frac{y(x_i+h)-y(x_i-h)}{2h} = \frac{y_{i+1}-y_{i-1}}{2h} = f(x_i,y_i)\ . \tag{6}$$

Hitro se lahko prepričamo, da je napaka sorazmerna s h^3. Metoda je natančnejša od Eulerjeve, vendar zelo *nestabilna* in zato praktično neuporabna.

Za resno delo namesto Eulerjeve metode izberemo boljšo metodo, na primer metodo Runge-Kutta četrtega reda. Zgled za uporabo metode predstavimo v Dodatku. Program `RK4` premakne rešitev (n-terico iskanih funkcij) za en korak naprej: $y_i(x) \to y_i(x+h),\ i = 1,\ldots,n$. Na vhodu moramo podati vektor vrednosti odvisnih spremeljivk, (`y[1..n]`) in njihove odvode (`dydx[1..n]`) pri x, število spremenljivk (`n`), vrednost neodvisne spremeljivke (`x`) in korak (`h`); na izhodu dobimo vrednosti (`yout [1..n]`) pri $x+h$. Glavnemu programu, ki kliče proceduro `RK4`, moramo dodati še proceduro `DERIVS`, ki izračuna odvode v trenutni točki. Vhod te procedure je vrednost neodvisne spremenljivke `x` in vektor vrednosti neodvisne spremenljivke `y[1..n]`; izhod pa vektor odvodov `dydx[1..n]`. Proceduro `RK4` torej kličemo v zanki; pred klicem moramo izračunati odvode v začetni točki (pokličemo proceduro `DERIVS`).

Enačbe prvega reda srečamo v fiziki pri problemih segrevanja (ohlajanja), praznenju kondenzatorja, absorpcije, radioaktivnih razpadov ...

Obdelajmo primer časovne odvisnosti temperature majhnega, dobro prevodnega telesa, ko lahko zanemarimo krajevno odvisnost temperature: diferencialno enačbo dobimo iz energijskega zakona. Vzemimo primer žarilne nitke, ki jo grejemo s tokom (gl. knjigo Matematične naloge iz fizike), le da bomo namesto izmeničnega toka uporabili tokovni sunek iz kondenzatorja:

$$mc\cdot\frac{dT}{dt} = RI_0^2\cdot\exp\left(-\frac{2t}{RC}\right) - \sigma S T^4\ . \tag{7}$$

Z vpeljavo brezdimenzijskih spremenljivk dobimo enoparametrično enačbo

$$\frac{dy}{dx} = a\cdot\exp(-2x) - y^4\ . \tag{8}$$

Zanemarili smo sevalni tok, ki ga nitka prejema iz okolice, zato se bo temperatura po dovolj dolgem času poljubno približala absolutni ničli: to pomeni, da rešitve ne kaže vleči predolgo. Da se izognemo

tudi preračunavanju začetne temperature, nas bodo zanimali samo zelo močni tokovni sunki, ko lahko začetno notranjo energijo nitke zanemarimo. K enačbi (8) velja torej začetni pogoj $u(x = 0) = 0$.

Najpreprostejši program v Pascalu je videti takole:

```
program Nitka;

VAR
     xn, x, y, h, a : Real;
     i, n           : Integer;

BEGIN
     a  := 4;
     xn := 2;
     y  := 0;
     write('Vtipkaj korak h >');
     Readln(h);
     n := round(xn / h) + 1;
     FOR i:= 1 TO n DO BEGIN
        x := h * i;
        y := y +  h * (a*exp(-2*x) - sqr(sqr(y)));
        Writeln( x:9:3 , y:9:4 )
     END {FOR};
     Readln;
END {Nitka}.
```

Neodvisna spremenljivka x teče med 0 in $x_n = 2$. Število delitvenih točk (n) izračunamo iz vrednosti koraka h, ki ga preberemo. Ker zaporednih točk ne shranjujemo, vrednosti za x in y pišemo kar preko starih vrednosti. Če jih želimo shraniti v polji $x[1..n+1]$ in $y[1..n+1]$, napišemo:

```
y[1] := 0;
x[1] := 0;
FOR i := 1 TO n DO BEGIN
   x[i+1] := h * i;
   y[i+1] := y[i] + h * (a*exp(-2*x[i]) - sqr(sqr(y[i])))
END;
```

Še prej moramo v glavi programa definirati obe polji. Rezultate je lepše podati v grafični obliki; zgled za uporabo grafike v Turbo Pascalu je predstavljen v Dodatku.

1.21 *Naloge*

1. Pri zgledu z nitko ugotovi, kolikšen korak je potreben za dovolj natačno računanje (na primer na 1 %).

2. Program priredi tako, da poišče in izpiše maksimalno temperaturo in čas, ko nastopi.

3. Izračunaj in grafično prikaži družino rešitev za različne a.

4. Pri zgledu uporabi še metodo Runge Kutta.

1.3 Enačbe drugega reda (Newtonov zakon)

1.31 *Gibanje v eni razsežnosti*

Če se omejimo na gibanje vzdolž ene same koordinate x, opiše gibanje masne točke v polju sil enačba drugega reda:

$$\frac{d^2x}{dt^2} = \frac{F}{m} = f(t,x,v)\,. \tag{9}$$

Enačba je enakovredna sistemu enačb prvega reda

$$\frac{dx}{dt} = v\,, \qquad \frac{dv}{dt} = f(t,x,v)\,. \tag{10}$$

Uporabimo lahko Eulerjevo shemo (6), še raje pa metodo Runge Kutta, saj nas pogosto zanima obnašanje sistemov tudi pri daljših časih.

Kadar sila F ni odvisna od hitrosti, obstaja za gornji sistem poleg standardnih metod tudi posebna trapezna shema, ki je presenetljivo točna in periodično stabilna

$$\begin{aligned} x(t+h) &= x(t) + h\,u(t+\tfrac{1}{2}h)\,, \\ u(t+\tfrac{1}{2}h) &= u(t-\tfrac{1}{2}h) + h\,f(x(t),t)\,. \end{aligned} \tag{11}$$

Pomožna spremenljivka u je le slaba aproksimacija za hitrost, še najbliže ji je v vmesnih točkah $(h/2)$, zato jo tja tudi pišemo. V resnici je u zgolj prva diferenca x. Sistem (11) je pravzaprav ekvivalenten shemi, ki jo dobimo, če drugi odvod v (9) aproksimiramo z

$$\frac{x(t+h) - 2x(t) + x(t-h)}{h^2} = f(t,x)\,, \tag{12}$$

od koder dobimo

$$x_{i+1} = 2x_i - x_{i-1} + h^2\, f_i \qquad f_i \equiv f(t_i,x_i)\,. \tag{13}$$

Sistemu (13) je mogoče še zvišati red od $O(h^3)$ na $O(h^5)$, če členu f dodamo še 1/12 njegove druge diference (metoda *Numerova*):

$$x_{i+1} = 2x_i - x_{i-1} + \frac{h^2}{12}\,(f_{i+1} + 10\,f_i + f_{i-1}) \tag{14}$$

Shema sedaj ni več eksplicitna, saj je f_{i+1} lahko nelinearno odvisen od x_{i+1}. Pomagamo si z iteracijo, tako da drugo diferenco najprej ocenimo, nato pa zares izračunamo.

Velja še opozoriti, da pri zagonu sheme (11) ali (14) potrebujemo vrednost rešitve ob času $h/2$ ($v(h/2)$ oziroma h ($x(h)$). (Začetni pogoj podaja odvod (hitrost) le ob času $t = 0$.) Najlažje si pomagamo tako, da rešitev razvijemo po Taylorju okoli $t = 0$; pri tem moramo vzeti vsaj toliko členov, kolikor je red metode. V izrazu

$$x(h) = x(0) + h\,x'(0) + \tfrac{1}{2}h^2\,x''(0) + \ldots + \tfrac{1}{n!}\,h^n\,x^{(n)}(0) + \ldots \tag{15}$$

dobimo $x(0)$ in $x'(0)$ iz začetnega pogoja, višje odvode pa izračunamo takole:

$$x^{(n)}(0) = f^{(n-2)} = \mathcal{F}\,(t,x(0),x'(0),\ldots,x^{(n-2)}(0))\,. \tag{16}$$

1.32 *Naloge*

1. Pri problemu dušenega nihanja nariši potek kinetične, prožnostne in celotne energije nihala. Preveri, če celotna energija pada kot $\mathrm{e}^{-2\beta t}$. Določi odstopanja.

2. Obdelaj matematično nihalo, tako da člen $\sin\phi$ obravnavaš točno, brez običajne aproksimacije za majhne kote.

1.33 *Gibanje v dveh razsežnostih*

V splošnem opiše gibanje masne točke v polju sil diferencialna enačba drugega reda

$$\frac{d^2\mathbf{r}}{dt^2} = \frac{\mathbf{F}}{m} = \mathbf{f}(t,\mathbf{r},\mathbf{v}) \,, \tag{17}$$

ki je enakovredna sistemu enačb prvega reda

$$\frac{d\mathbf{r}}{dt} = \mathbf{v} \,, \qquad \frac{d\mathbf{v}}{dt} = \mathbf{f} \,. \tag{18}$$

Vektorski zapis seveda pomeni, da imamo opravka s (sklopljenim) sistemom enačb za vsako komponento vektorja $\mathbf{r}$ posebej.

Najpreprostejši primer je poševni met. Pri numeričnem reševanju lahko brez težav vključimo upor zraka, ki je običajno odvisen od kvadrata hitrosti telesa.

Še bolj zanimiv in zahteven je izračun trajektorije satelita, ki se giblje v gravitacijskem polju, na primer Zemlje in Lune. Zapišimo jih za nekoliko poenostavljen primer, ko ne upoštevamo kroženja Zemlje in Lune okoli skupnega težišča. Velja

$$\frac{d^2\mathbf{r}}{dt^2} = -\frac{\kappa M_Z \mathbf{r}}{r^3} - \frac{\kappa M_L \mathbf{r}'}{r'^3} \,, \qquad \mathbf{r}' = \mathbf{r} - \mathbf{r}_{ZL} \,, \tag{19}$$

če je M_Z masa Zemlje, M_L masa Lune in $\mathbf{r}_{ZL}$ vektor, ki sega od središča Zemlje do središča Lune. Z izbiro novih spremeljivk (razdalje merimo v Zemljinih polmerih R_Z, hitrosti pa s hitrostjo kroženja satelita tik nad površjem Zemlje $v_0 = \sqrt{gR_Z} = \sqrt{\kappa M_Z / R_Z}$) dobimo sistem enačb:

$$\begin{aligned}
\frac{dx}{dt} &= u \,, \\
\frac{du}{dt} &= -\frac{x}{\sqrt{x^2+y^2}^3} - \gamma \frac{(x - r_{ZL})}{\sqrt{(x-r_{ZL})^2+y^2}^3} \,, \\
\frac{dy}{dt} &= v \,, \\
\frac{dv}{dt} &= -\frac{y}{\sqrt{x^2+y^2}^3} - \gamma \frac{y}{\sqrt{(x-r_{ZL})^2+y^2}^3} \,.
\end{aligned} \tag{20}$$

Enačb ni težko razširiti tako, da upoštevamo še kroženje Zemlje in Lune okoli skupnega težišča. (Če se postavimo v vrteči se sistem, moramo vključiti še centrifugalno in Coriolisovo silo.) V primeru statičnega polja se ohranja vsota kinetične in potencialne energije; to pa ni več res v realističnem primeru. Tedaj lahko pride do zanimivega pojava, ko sistem obeh planetov izstreli satelit izven območja svoje težnosti, čeprav je začetna hitrost satelita manjša od druge kozmične hitrosti. Do takšnega pojava lahko sicer pride tudi zaradi numeričnih nestabilnosti v statičnem primeru, ko se satelit preveč približa izvoru gravitacijskega polja. Zato je smiselno vpeljati končne razsežnosti planetov in računanje prekiniti, ko satelit trešči na površje planeta.

Pri resnih računih te vrste izberemo metodo Runge Kutta s spremenljivim korakom. Korak se avtomatično poveča, ko se satelit giblje po pohlevni krivulji daleč proč od planetov, in se zmanjša, ko pride v bližino močnega gravitacijskega polja.

1.34 *Naloge*

1. Zapiši enačbe za poševni met z upoštevanjem upora zraka in jih reši z Eulerjevo metodo. Razišči, kako se spreminja optimalni kot meta v odvisnosti od koeficienta upora. (Najprej preveri,

če dobiš pravilno rešitev v primeru, ko ni upora. Tudi tu velja paziti, da bo imela zaviralna sila vedno nasprotno smer hitrosti. Komponento v vodoravni smeri napišemo kot $F_x = -b\,|v|\,v_x = -b\sqrt{v_x^2 + v_y^2}\,v_x$.)

1.4 Zgled: Tiri satelitov z metodo Runge-Kutta

```
Program Orbite;

USES Graph, Crt;

CONST
      Pot = 'C:\TP\BGI'; h = 0.1;  visina = 0.2;
TYPE
      glnarray = ARRAY [1..4] OF real;
VAR
      GD, GM, ix_max, iy_max, i, N , s_x, s_y, ix, iy, j : Integer;
      x, v_0 , f_x, f_y, r, r3                           : Real;
      y, yout,dydx                                       : glnarray;

PROCEDURE VkljuciGrafiko;

BEGIN
    DetectGraph(GD, GM);
    InitGraph(GD, GM, Pot);
    ix_max := GetMaxX;
    iy_max := GetMaxY;
    s_x := iy_max DIV 4;
    s_y := iy_max DIV 2 ;
    SetBkColor(Blue);
    f_y:= 0.1 * iy_max;
    f_x:= 0.1 * iy_max;
    SetFillStyle( 1 , Green );
    FillEllipse( s_x, s_y , round(f_x), round(f_x) ) ;
END{VkljuciGrafiko};
```

```
PROCEDURE derivs(x:real; y:glnarray; VAR dydx:glnarray);

BEGIN
   r    := sqrt( sqr(y[1]) + sqr(y[2]));
   r3   := r * r * r;
   dydx[1] := y[3];              { dx / dt = v_x }
   dydx[2] := y[4];              { dy / dt = v_y }
   dydx[3] := - y[1] / r3;       { dv_x / dt = F_x = - x / r^3 }
   dydx[4] := - y[2] / r3;       { dv_y / dt = F_y = - y / r^3 }
END{derivs};

PROCEDURE rk4(y,dydx: glnarray; n: integer; x,h: real; VAR yout: glnarray);

...

END{rk4};

BEGIN{orbite}
   VkljuciGrafiko;
   REPEAT
     SetColor(White);
     OutTextXY(5,5,'hitrost [.8 - 1.2, 0=stop] =');
     GotoXY(40,1); Read(v_0);
     SetColor( LightRed );
     IF v_0 > 0 THEN BEGIN
        y[1] := - (1 + visina); { x(0) }
        y[3] :=   0;            { v_x(0) }
        y[2] :=   0;            { y(0) }
        y[4] := v_0;            { v_y(0) }
        N  := round(100 / h) + 1;
        x := 0;
        ix := round(y[1]*f_x) + s_x;
        iy := round(y[2]*f_y) + s_y;
        MoveTo(ix, iy);
        FOR i:= 1 TO N DO BEGIN
           derivs(x,y,dydx);
           rk4(y,dydx,4,x,h,y);
           x := x + h;
           ix := round(y[1]*f_x) + s_x;
           iy := round(y[2]*f_y) + s_y;
           LineTo(ix,iy);
        END {FOR};
      END {IF}
    UNTIL v_0 <= 0;
    CloseGraph;
END {orbite}.
```

2. Numerični izračun Fourierove transformacije

Pri numeričnem izračunavanju Fourierove transformacije

$$A_n = \frac{1}{T}\int_0^T h(t)\exp(2\pi \mathrm{i} n\nu t)\,dt\ , \qquad \nu = \frac{1}{T}\ , \tag{21}$$

$$h(t) = \sum_{n=-\infty}^{\infty} A_n \exp(-2\pi \mathrm{i} n\nu t) = A_0 + \sum_{n=1}^{\infty}(A_n + A_{-n})\cos 2\pi n\nu t - \mathrm{i}(A_n - A_{-n})\sin 2\pi n\nu t \tag{22}$$

je funkcija $h(t)$ običajno predstavljena s tablico diskretnih vrednosti

$$h_k = h(t_k), \quad t_k = k\Delta\ , \qquad k = 0,1,2,\ldots N-1\ . \tag{23}$$

Pravimo, da smo funkcijo vzorčili z vzorčno gostoto $1/\Delta$. Za tako definiran vzorcu obstaja naravna meja frekvenčnega spektra, ki se imenuje *Nyquistova frekvenca*, $f_c = 1/(2\Delta)$: harmonični val s to frekvenco ima v naši vzorčni gostoti ravno dva vzorca v periodi. Če ima funkcija $h(t)$ frekvenčni spekter omejen na interval $[-f_c, f_c]$, potem ji z vzorčenjem nismo odvzeli nič informacije: kadar pa se spekter razteza izven intervala, pride do *potujitve* (*aliasing*), ko se zunanji del spektra preslika v interval.

Frekvenčni spekter vzorčene funkcije (23) spet računamo samo v N točkah, če hočemo, da se ohrani količina informacije. Vpeljemo vsoto

$$H_n = \sum_{k=0}^{N-1} h_k \exp(2\pi \mathrm{i} kn/N)\ , \qquad n = -N/2,\ldots,N/2, \tag{24}$$

ki jo imenujemo diskretna Fourierova transformacija, in je povezana s funkcijo v (21) takole:

$$A_n = \frac{\Delta}{T}H_n = \frac{1}{N}H_n\ . \tag{25}$$

Zaradi potujitve, po kateri je $H_{-n} = H_{N-n}$, lahko mirno pustimo indeks n v enačbi (24) teči tudi od 0 do N. Spodnja polovica tako definiranega spektra ustreza pozitivnim frekvencam med 0 in f_c , gornja pa negativnim od $-f_c$ do 0.

Z diskretno obratno transformacijo lahko rekonstruiramo h_k iz H_n

$$h_k = \frac{1}{N}\sum_{n=0}^{N-1} H_n \exp(-2\pi \mathrm{i} kn/N)\ . \tag{26}$$

Količine h in H so v splošnem kompleksne, simetrija v enih povzroči tudi simetrijo v drugih. Za realne h so H paroma konjugirani, $H_{-n} = H_n^*$ in velja:

$$a_n \equiv A_n + A_{-n} = 2\Re A_n = \frac{2}{N}\Re H_n\ , \quad b_n \equiv -\mathrm{i}(A_n - A_{-n}) = 2\Im A_n = \frac{2}{N}\Im H_n\ . \tag{27}$$

2.1 Hitra Fourierova transformacija (FFT)

Postopek (24) ima očitno časovno zahtevnost N^2 . Račun pa je mogoče izvesti tudi z $N\cdot\log_2 N$ množenji. Osnovni premislek je naslednji razcep

$$F_m = F_{m^e} + W^m F_{m^o}\ , \qquad W = \exp(2\pi \mathrm{i}/N) \tag{28}$$

kjer smo transformiranko F izrazili s transformacijama njenih sodih in lihih členov. Gornjo relacijo lahko uporabljamo rekurzivno: če je N enak potenci števila 2, lahko rekurzijo razdrobimo do nizov, ki imajo samo še en člen. Zanj je transformacija identiteta. Za obrat pri eni vrednosti frekvence (pri danem m) je potrebno na vsakem koraku rekurzije le eno množenje s potenco W, korakov pa je je $\log_2 N$.

Da ne iščemo pripadnikov niza po vsej tabeli, si podatke preuredimo. Lahko je pokazati, da je v prvotni tabeli treba med seboj zamenjati podatke, katerih vrstna števila v binarnem zapisu so obrnjena: v novem redu jemljemo člene kar po vrsti. Tudi potenc W ne izražamo vedno znova s sinusi in kosinusi, pač pa jih računamo z rekurzijo.

Uporabimo proceduro `four1` iz zbirke NUMERICAL RECIPES. *Vhodni parametri* procedure so polje `data[2*nn]` z elementi `data[1]` $= \Re h_0$, `data[2]` $= \Im h_0$, `data[3]` $= \Re h_1 \ldots$, `nn` je število podatkov (N) in mora biti enako potenci števila 2. Parameter `isign` je enak 1 pri Fourierova transformacija in -1 pri obratni Fourierovi transformaciji; *izhodni parametri:* transformiranka "povozi" podatkovno polje `data[2*nn]` in velja `data[1]` $= N \cdot \Re H_0$, `data[2]` $= N \cdot \Im H_0$, `data[3]` $= N \cdot \Re H_1 \ldots$.

2.11 *Naloge*

1. Generiraj nekaj signalov, ki predstavljajo linearno superpozicijo valovanj s frekvencami, ki so mnogokratniki osnovne frekvence. Preveri, da z obratno transformacijo dobiš prvotni niz.

2. Analiziraj še signale, katerih frekvence niso mnogokratniki osnovne frekvence. Kako dolžina niza vpliva na obliko transformiranke?

3. Osnove numeričnega reševanja parcialnih diferencialnih enačb

3.1 Difuzijska enačba — Fourierova metoda

Temperaturno polje v homogeni neskončni plasti s končno debelino a (enodimenzionalni problem) je podano z enačbo

$$\frac{\partial T}{\partial t} = D\frac{\partial^2 T}{\partial x^2} + \frac{q}{\rho c}\,, \qquad 0 < x < a, \qquad D = \frac{\lambda}{\rho c}\,. \tag{29}$$

Iz začetne temperaturne slike $T(x,t=0) = G(x)$ lahko s pomočjo lastnih funkcij $u_i(x)$ rešitev takoj zapišemo

$$T(x,t) = \sum_i [A_i + (B_i - A_i)\exp(-Dk_i^2 t)]\, u_i(x)\,, \tag{30}$$

kjer so koeficienti A in B podani z

$$A_i = -\frac{1}{\lambda}\frac{(q(x),u_i)}{k_i^2}\,, \qquad B_i = (G,u_i)\,, \tag{31}$$

kjer zapis (f,u_i) pomeni $(f,u_i) = \int_0^a f(x)u_i(x)\,dx$. Lastne funkcije u so podane šele z robnima pogojema, ki morata biti linearna in celo homogena. V posebnih primerih pogojev I. in II. vrste je mogoče račun izvesti z učinkovitim algoritmom FFT.

3.2 Valovna enačba — Fourierova metoda

Podobno velja za začetni problem pri valovni enačbi

$$\frac{\partial^2 z}{\partial t^2} = c^2\frac{\partial^2 z}{\partial x^2}\,, \qquad 0 < x < a, \tag{32}$$

kjer sta predpisana začetni odmik $z(x,t=0) = f(x)$ in začetna hitrost $\partial z/\partial t|_{t=0} = g(x)$. Rešitev se dobi

$$z(x,t) = \sum_i \left[(f,u_i)\cos\omega_i t + \frac{(g,u_i)}{\omega_i}\sin\omega_i t\right] u_i(x)\,. \tag{33}$$

Če izberemo robna pogoja I. vrste na obeh krajiščih intervala, tako da so lastne funkcije sinusi z zaporednimi števili polvalov na intervalu. FFT pomeni sicer razvoj po drugačnih funkcijah, vendar lahko uporabimo preprosto zvijačo: funkcijo, ki jo želimo razviti, podaljšamo v liho funkcijo na dvojnem intervalu.

3.21 *Naloge*

1. Zasleduj časovni razvoj temperaturnega profila v plasti, ki ima v začetku povsod temperaturo okolice, le med $0.2a$ in $0.3a$ je segreta na T_0. Nariši $T(x,t)$ ob $Dt/a^2 = 0$ (0.3) 3!

2. 2. Zasleduj časovni razvoj oblike strune, ki ima v začetku trikotno obliko z vrhom pri $x = 0.4a$, za čase $\omega_1 t/\pi = 0$ (0.2) 2! Začetna hitrost naj bo povsod nič.

3.3 Difuzijska enačba — diferenčna metoda

Diferenčne metode za aproksimativno reševanje parcialnih diferencialnih enačb so široko uporabne, saj niso omejene na preproste geometrije in linearne robne pogoje. Pri aproksimaciji odvodov s končnimi diferencami se običajno zadovoljimo z najnižjim možnim redom, formule višjega reda so zelo rade nestabilne.

Zaradi primerjave bomo obdelali oba zgornja primera. Za difuzijsko enačbo zapišemo najpreprostejšo aproksimacijo

$$\frac{u_{i+1,j}-u_{i,j}}{k} = D\cdot\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{h^2}+Q, \tag{34}$$

kjer štejemo z indeksom i časovne sloje v razmikih $\Delta t = k$, z indeksom j pa označimo krajevne točke, $0 \le j \le N$ v enem sloju. Ob času $t = 0$ je z začetnim pogojem podan prvi sloj, $i = 0$, iz njega izračunamo drugi sloj, in tako naprej. Enačba (34) velja za vse notranje točke, obe robni točki pa določata robna pogoja. Ta eksplicitna shema je preprosto izvedljiva in stabilna za $p = Dk/h^2 \le 1/2$, ni pa posebno točna, saj je v časovnem odvodu le prvega reda. Nekoliko točnejša postane za $p = 1/6$, vendar napredujemo pri tem koraku zelo počasi, saj predifundira rešitev šele v $6N^2$ korakih od enega konca krajevnega intervala do drugega.

Boljša je *Crank-Nicholsonova* shema, ki je drugega reda v času: časovno diferenco na levi strani enačbe (34) izrazimo z aritmetično sredino krajevnih diferenc v sloju i in sloju $i+1$. Novih vrednosti v sloju $i+1$ ne dobimo eksplicitno, pač pa kot rešitve tridiagonalnega sistema enačb. Ker je časovna zahtevnost takega sistema le $\sim N$, se z računom ne zamudimo dosti dlje, stabilnost pa je zagotovljena za poljubno velik korak. Ker rešujemo enačbo s konstantnimi koeficienti, je tudi matrika sistema vedno ista, spreminjajo se le desne strani:

$$(-pu_{j+1}+(2p+2)u_j-pu_{j-1})_{i+1} = (pu_{j+1}-(2p-2)u_j+pu_{j-1})_i+2kQ. \tag{35}$$

3.4 Valovna enačba — diferenčna metoda

Za valovno enačbo obstaja stabilna eksplicitna metoda drugega reda v času. Dobimo jo, če v enačbi (32) zapišemo časovni operator na levi strani kot

$$\frac{(u_{i+1,j}-2u_{i,j}+u_{i-1,j})}{k^2}\,, \tag{36}$$

operator na desni pa tako kot v (34). Vrednosti na časovnem sloju $i+1$ torej izrazimo s slojema i in $i-1$. Začetna sloja dobimo iz dveh začetnih pogojev: za funkcijo in za njen časovni odvod. V posebnem primeru, ko so začetne hitrosti povsod enake 0, morata biti sloja $i = 1$ in $i = -1$ enaka in velja

$$u_{1,j} = u_{0,j}+\tfrac{1}{2}\cdot\left(\frac{kc}{h}\right)^2\cdot(u_{0,j+1}-2u_{0,j}+u_{0,j-1})\,. \tag{37}$$

Metoda je stabilna za $kc/h \le 1$, kar pomeni, da zadošča $\sim N$ korakov za pot vala od enega konca intervala do drugega. Večji korak omogočajo implicitne metode, vendar z njimi ne moremo pridobiti dosti.

3.41 *Naloge*

1. Obe predhodni nalogi reši še z diferenčno metodo in primerjaj njeno učinkovitost s Fourierovo metodo.

4. Metode Monte Carlo

Metoda Monte Carlo je bila razvita predvsem za računanje večkratnih integralov, kakršne pogosto srečamo v statistični in kvantni fiziki. Vendar to še ne pomeni, da metoda spada med suhoparna poglavja matematične fizike ali numerične analize. Nasprotno, pogosto lahko postopek ponazorimo s procesom, ki ima zanimivo fizikalno vsebino: računi Monte Carlo in računalniške simulacije fizikalnih procesov so med seboj tesno prepleteni in povezani. Čeprav se metoda uporablja na fronti fizikalnih raziskav, je njeno ozadje razmeroma preprosto in ga lahko razložimo na elementarnem nivoju. S pomočjo metode Monte Carlo lahko tudi pripravimo zanimive računalniške simulacije fizikalnih procesov.

4.1 Osnove integracije Monte Carlo

Osnovno idejo pojasnimo na zgledu računanja večkratnega integrala – na primer ploščine lika v ravnini. Na najpreprostejši način izračunamo ploščino tako, da lik pokrijemo z mrežo (recimo milimetrsko) in preštejemo število kvadratkov znotraj lika. Čim gostejšo mrežo izberemo, tem natačnejši bo rezultat. Namesto z mrežo lahko lik pokrijemo z enakomerno gosto množico točk in preštejemo tiste, ki se znajdejo znotraj lika. Rezultat umerimo tako, da preštejemo število točk znotraj lika z znano ploščino. Kako dobimo primerno množico točk? Preprosto: točke postavljamo na enakih razdaljah drugo od druge. Pa je res nujno, da so razdalje enakomerne? Metoda Monte Carlo pravi da ne; lik lahko pokrijemo tudi z množico naključno raztresenih točk, pomembno je le, da pri tem nobenega dela prostora (ravnine) ne priviligiramo.

Povejmo kar konkreten primer. Število $\pi/4$ lahko dobimo tako, da poiščemo ploščino kroga, včrtanega kvadratu s stranico 1. Z generatorjem naključnih števil generiramo točke znotraj kvadrata; če je generator dober in če izžrebamo dovolj veliko število točk (dvojic naključnih števil), bo kvadrat pokrit zadovoljivo enakomerno in ploščini kvadrata in včrtanega kroga bosta približno v razmerju celotnega števila točk, N, in števila točk, M, ki padejo znotraj kvadrata. Približek za število π potemtakem izrazimo kot $4 \cdot M/N$. Postopek je seveda potraten, saj obstaja za računanje števila π mnogo učinkovitejših metod.

Seveda pa obstajajo primeri, ko je opisani postopek ekonomičnejši od drugih. Da to pokažemo, si moramo najprej ogledati, kako je z napako pri metodi Monte Carlo? Pri tej metodi integral

$$I = \int_a^b f(x)\,dx \tag{38}$$

nadomestimo z vsoto

$$I \approx (a-b)\,\frac{1}{N}\sum_{i=1}^{N} f(x_i) = (a-b)\,\frac{1}{N}\sum_{i=1}^{N} f_i = (a-b)\,\overline{f}\,, \tag{39}$$

če so $x_i, i = 1,\ldots N$ naključno izbrana števila z intervala $[a,b]$. Tudi funkcijske vrednosti $f_i \equiv f(x_i)$ so porazdeljen naključno. Integral je torej enak širini intervala, pomnoženi s povprečno vrednostjo funkcije. Napaka, ki jo zagrešimo pri integriranju z metodo Monte Carlo, je torej sorazmerna z napako pri računanju povprečja funkcijski vrednosti f_i. Naj bo σ_f napaka, ki jo v povprečju zagrešimo, če za povprečno vrednost funkcije vzamemo nakjučno izbrano vrednost $f(x_i)$. Napaka je očitno tem večja, čim bolj se funkcija spreminja znotraj intervala. Iz statistike vemo, da se napaka manjša s številom poskusov (‘meritev’) takole:

$$\sigma_f^2(N) = \frac{1}{N}\,\sigma_f^2 \tag{40}$$

torej je napaka integrala

$$\sigma_I = (a-b)\,\sigma_f(N) = (a-b)\,\frac{1}{\sqrt{N}}\,\sigma_f\ . \tag{41}$$

Napaka se manjša z večanjem števila poskusnih točk s korensko odvisnostjo. Če funkcija ni preveč enakomerna, ocenimo, da za relativno natačnost 1 % potrebujem 10000 točk.

Pri računanju integralov v eni dimenziji je metoda hudo potratna, saj lahko že z najpreprostejšo kvadraturno formula izračunamo integral z natačnostjo $1/N^2$, če je N število izvrednotenj funkcije (število delitev intervala). Pri integraciji v več dimenzijah to ni več res. Tu razdelimo interval na primer z N_1 točkami v vsaki dimenziji, kar pomeni, da moramo funkcijo izračunti N_1^d krat, če je d število dimenzij. Ker je napaka sorazmerna s širino delitvenega intervala, velja:

$$\sigma_I \propto h_1^2 \propto N_1^{-2} = N^{-2/d}\ . \tag{42}$$

Pri računanju večkratnih integralov z metodo Monte Carlo ostane še vedno v veljavi ocena za napako $\sigma_I \propto N^{-1/2}$, saj pri izpeljavi nismo nikjer predpostavili, da gre za enodimenzionalni integral. Primerjava obeh ocen pokaže, da je metoda Monte Carlo uspešnejša od običajnih metod, če je dimenzija prostora večja od 4.

Prvi hip se nam zazdi, da integralov v več kot treh dimenzijah v praksi sploh ne srečamo. Vendar ni nujno, da je integracijski prostor navadni prostor: v statistični fiziki običajno zapišemo integrale v *faznem* prostoru – poleg po koordinatah delca integriramo še po treh komponentah njegove gibalne količine. Sistem n delcev tako opisujemo v abstraktnem $6n$-dimenzionalnem prostoru in če želimo numerično izvrednotiti konkreten integral, nam ostane na razpolago le metoda Monte Carlo. Podobno je v kvantni mehaniki, le da je tu prostor sistema n delcev $3n$ dimenzionalen.

4.2 Generator naključnih števil

Pomemben element metode Monte Carlo je naključje, saj želimo generirati integracijske točke čim bolj naključno, podobno kot nam narava naključno poseje merske vrednosti okoli povprečja. Od tod seveda tudi ime metode, ki je aluzija na naključno generirana števila na ruleti. Osnovno orodje, ki ga potrebujemo na računalniku, je generator naključnih števil. Pa lahko naprava, ki je *par excellance* deterministična, sploh generira kakršno koli naključnost? Seveda ne; lahko pa dobimo zaporedje števil, ki dovolj dobro igra vlogo naključnih števil za namen, za katerega jih nameravamo uporabljati. Takšnim številom pravimo *psevdonaključna števila*.

Najbolj znan je *kongruenčni* algoritem, pri katerem dobimo novo naključno število x_{i+1} iz starega x_i takole:

$$x_{i+1} = (a\,x_i + c) \bmod m \tag{43}$$

Števila a, c in m so posebno izbrana števila. Iz formule hitro uvidimo, kdaj postopek odpove: če generiramo novo število, ki se je že pojavilo v zaporedju, se od tam naprej zaporedje ponovi in generator ni več uporaben. S primerno izbiro števil a, c in m torej poskrbimo, da je tak cikel čim večji. Dolžino cikla lahko bistveno povečamo, če uporabimo več kongruenčnih algoritmov hkrati.

Pred resno uporabo velja generator temeljito preskusiti. Poleg enakomernosti moramo vsaj še preveriti, če točke niso med seboj korelirane.

Fizikalno zanimiv je preskus z *naključno hojo*. Sprehajalec se pri vsakem koraku naključno odloči, v katero smer se bo napotil. Če je generator, s katerim izžrebamo smer, dober, se sprehajalec obnaša kot Brownov delec. Tak delec difundira v sredstvu, za takšno gibanje pa velja, da se kvadrat razdalje od izhodišča povečuje linearno s časom:

$$R(t)^2 \propto t\ . \tag{44}$$

Napišimo program, ki opisuje hkratno gibanje večjega števila sprehajalcev – ali pa difuzijo delcev v sredstvu. Zaradi enostavnosti se omejim na hojo po dvodimenzionalni mreži; na vsakem koraku naj sprehajalec naključno stopi naprej, nazaj, levo ali desno. Uporabimo kar generator naključnih števil, ki je vgrajen v TURBO PASCAL: z `npts` preberemo število sprehajalcev, z `nstep` pa število korakov:

```
program Diffuz;
uses Graph;
const Pot = 'C:\TP\BGI';
var      sqrd, oldsqrd : Longint;
var      w , d2 , dd2 , pd2: Real;
var      GD, GM, npts, nsteps, maxy : Integer;
var      i, j, k, smer, ix0, iy0, x2, y2  : Integer;
var      ix  : array [0..10000] of Integer;
var      iy  : array [0..10000] of Integer;
var      ImeN , Ime , Imed , Imep : String;
BEGIN
    w := random;         {Po"zenemo generator naklju"cnih stevil}
    REPEAT
    write('npts=? (<10000) >');
    Readln(npts);               {Preberemo "stevilo sprehajalcev}
    IF npts > 0 THEN BEGIN
    write('nsteps = 25*k, izberi k [10]  >');
    Readln(nsteps);                        {Preberemo "stevilo korakov}
{       nsteps := nsteps DIV 25; }
       DetectGraph(GD, GM);
       InitGraph(GD, GM, Pot);             {Sko"cimo v grafi"cni na"cin}
       maxy := GetMaxY - 20;
       ix0 := maxy DIV 2; iy0 := ix0;      {Dolo"cimo izhodi"s"ce}
       FOR i:=1 TO npts DO BEGIN
         ix[i] := ix0; iy[i] := iy0;       {Vsi pri"cno v izhodi"s"cu}
       END;
       SetFillStyle(1, Yellow);
       Bar(10,10,maxy+10,maxy+10);
       OutTextXY(maxy+10,20,'korak   <d^2>  d<d^2>');
       oldsqrd := 0;
       FOR k:=1 TO 25 DO BEGIN       {Vmesni rezultat izpi"semo 25 krat}
         FOR j:=1 TO nsteps DO BEGIN
           FOR i:=1 TO npts DO BEGIN
             PutPixel(ix[i],iy[i],Yellow);    {Zbri"semo sprehajalca}
             smer := trunc(4.*random);       {Iz"zrebamo 0,1,2 ali 3}
             CASE smer OF
               0 : ix[i] := ix[i] + 1;       {Korak v desno}
               1 : ix[i] := ix[i] - 1;       {levo}
               2 : iy[i] := iy[i] + 1;       {navzgor}
               3 : iy[i] := iy[i] - 1;       {navzdol}
             END;
             PutPixel(ix[i],iy[i],Magenta);    {Sprehajalec v novi to"cki}
           END;
```

```
        END;
        sqrd:= 0; {Izra"cunamo vsoto kvadratov razdalj od izhodi"s"ca}
        FOR i:=1 TO npts DO BEGIN
           x2 := sqr(ix[i]-ix0); y2 := sqr(iy[i]-iy0);
           sqrd := sqrd + x2 + y2;
        END;
        d2  := sqrd / npts;
        dd2 := (sqrd - oldsqrd) / npts;
        oldsqrd := sqrd;
        Str(k*nsteps,ImeN); Str(d2:8:1,Ime); Str(dd2:7:2,Imed);
        OutTextXY(maxy+10, 10*k+30,ImeN);
        OutTextXY(maxy+60, 10*k+30,Ime);        {Izpi"semo kvadrat razdalje}
        OutTextXY(maxy+120,10*k+30,Imed);       {in njegovo spremembo}
      END;
      pd2 := sqrd / 25 / nsteps / npts;
      Str(pd2:8:3,Imep);
      OutTextXY(maxy+10,300,'<sigma2/korak>=');
      OutTextXY(maxy+110,300,Imep);
  END;
  Readln;
  CloseGraph;
  UNTIL npts = 0
END.
```

4.3 Naključna števila po zahtevani porazdelitvi

Generatorji naključnih števil na računalnikih običajno dajo števila, ki so enakomerno porazdeljena na intervalu med 0 in 1. Pri metodi Monte Carlo pogosto potrebujemo naključna števila iz drugačnega intervala, pa tudi porazdeljena po porazdelitvi, ki ni enakomerna. Pri Gaussovi porazdelitvi na primer želimo najpogosteje dobiti naključno število v bližini predpisane vrednosti, števila, ki bi se precej razlikovala od te vrednosti, pa bolj poredko.

Enakomerno porazdelitev na poljubnem intervalu, denimo med a in b, dobimo z enostavno transformacijo:

$$\xi_i' = a + (b-a)\xi_i \,, \tag{45}$$

če je ξ_i naključno število iz intervala $(0,1)$.

Pri zgledu za računanje integralov v dveh razsežnostih smo navedli metodo, kako generiramo naključno število znotraj lika. Pri krogu lahko enakomerno porazdelitev dobimo tudi tako, da naključno generiramo polarni koordinati ρ in φ:

$$x_i = \rho_i \cos\varphi_i \,, \qquad y_i = \rho_i \sin\varphi_i \,. \tag{46}$$

Kar se tiče kota φ, je postopek enostaven, saj so vrednosti za kot enakomerno porazdeljene v intervalu $(0, 2\pi)$, torej:

$$\varphi_i = 2\pi\xi_i \,. \tag{47}$$

Pri koordinati ρ postopek ni tako enostaven, saj bi enakomerno generiranje na intervalu med 0 in radijem kroga, R, naključne točke zgoščevalo v bližini izhodišča. Seveda, zagotoviti moramo, da bo vsak del *površine* enakomerno pokrit; površine, ki pripadaju intervaly $d\rho$ pa naraščajo z rastočim ρ

kot $2\pi\rho\, d\rho$. Za gostoto naključnih točk mora veljati

$$\frac{dN}{dS} = \frac{dN}{2\pi\rho\, d\rho} = \text{konst.} \tag{48}$$

Da bomo lažje videli, kako je pogoj (48) povezan z generiranjem naključnih števil, se vrnimo na generiranje naključnih števil na intevalu med 0 in 1. Porazdelitvena funkcija je tu kar konstanta; iz pogoja, da je normirana na 1, sledi, da je njena višina enaka 1. Naključno število ξ, ki ga generamo, lahko potem predstavimo kot ploščino pravokotnika z višino 1 in osnovnico ξ, oziroma ploščino pod krivuljo, ki predstavlja verjetnostno gostoto, na intervalu $(0,\xi)$. S pomočjo takšne slike lahko rešitev za problem kroga zapišemo:

$$A\pi\rho_i^2 = \xi_i \,, \tag{49}$$

saj je $\pi\rho_i^2$ ploščina lika med 0 in naključnim številom ρ_i, ki ga želimo generirati. Konstanta A je izbrana tako, da je površina celotnega kroga enaka 1: $A = 1/\pi R^2$. Od tod razberemo:

$$\rho_i = R\sqrt{\xi_i} \,, \tag{50}$$

Iz zgornjega razmisleka hitro pridemo do splošnega recepta: če naj bo spremenljivka x porazdeljena po verjetnostni gostoti $dN/N_0 dx = w(x)$, potem dobimo nakjučno vrednost x_i kot

$$\int_0^{x_i} w(x)\, dx = \xi_i \tag{51}$$

(ξ_i je kot vedno naključno število iz intervala $(0,1)$). Formulo (51) lahko enostavno uporabimo le za porazdelitve, ki jih znamo integrirati in iz dobljenega integrala eksplicitno izraziti njegovo zgornjo mejo. (V primeru kroga je verjetnostna gostota $w(\rho) = 2\rho/R$.)

Za enakomerno porazdeljene točke znotraj krogle z radijem R brez težav uporabimo zgornji recept. V krogelnih koordinatah velja:

$$\frac{dV}{r^2\, dr\, \sin\theta\, d\theta\, d\phi} = \text{konstanta} \,. \tag{52}$$

Ker je

$$\frac{dV}{dr\, d\theta\, d\phi} = r^2\, \sin\theta = w(r)w(\theta)w(\phi) \,, \tag{53}$$

ugotovimo, da je pogoj (52) izpolnjen za

$$w(r) = \text{konstanta}'\, r^2 \,, \quad w(r) = \text{konstanta}''\, \sin\theta \,. \quad w(\phi) = \text{konstanta}''' \,, \tag{54}$$

kjer konstante določimo iz normalizacijskega pogoja. Tako dobimo:

$$r_i = R\sqrt[3]{\xi_i} \,, \tag{55}$$

$$\cos\theta_i = 2\xi_{i+1} - 1 \,, \tag{56}$$

$$\phi_i = 2\pi\xi_{i+2} \,, \tag{57}$$

če so ξ_i, ξ_{i+1} in ξ_{i+2} tri naključna števila med 0 in 1.

Povprečne proste poti molekul v plinu ali časi razpadov radioaktivnih jeder so porazdeljeni eksponentno: $w(x) = \mathrm{e}^{-x/a}$. Dobimo:

$$x_i = -a\ln(1-\xi_i) = -a\ln\xi_i' \,. \tag{58}$$

Drugi enačaj je upravičen, saj so naključna števila $1-\xi_i$ tudi porazdeljena enakomerno na intervalu med 0 in 1.

Gaussove porazdelitve ne moremo tako enostavno obrniti. Da pa se pokazati, da sta naključni števili x in y

$$x_i = \sqrt{-2\ln\xi_i}\cos 2\pi\xi_{i+1} \quad \text{in} \quad y_i = \sqrt{-2\ln\xi_i}\sin 2\pi\xi_{i+1} \tag{59}$$

porazdeljeni po Gaussovi porazdelitvi s širino 1 in vrhom pri 0.