БЕЛАРУСКІ ДЗЯРЖАЎНЫ УНІВЕРСІТЭТ

Факультэт прыкладной матэматыкі і інфарматыкі

Метады лікавага аналізу

Справаздача па лабараторнай працы
па тэме

Рознасныя схемы для ўраўненняў Пуасона і цеплаправоднасці

Богдана Уладзіслава
3 курс, 3 група

Выкладчык:
Буднік А. М.

Мінск, 2016

1 Задача Дырыхле для ўраўнення Пуасона

Пастаноўка задачы

Дадзеная задача Дырыхле для ўраўнення Пуасона на простакутніку $[a,b]\times[c,d]$:

$$\frac{\partial^2(u)}{\partial x^2}+\frac{\partial^2(u)}{\partial y^2}=-f(x,y), \tag{1}$$

$$x\in[a,b], y\in[c,d]$$

$$u(a,y)=\psi_1(y),\quad u(b,y)=\psi_2(y)$$

$$u(x,c)=\psi_3(x),\quad u(x,d)=\psi_4(x)$$

Ставіцца мэта знайсці набліжанае рашэнне рознасным ітэрацыйным метадам на сетцы вузлоў з крокамі $h_x=h_y=0.05$.

У маім выпадку задача мае наступны выгляд:

$$\frac{\partial^2(u)}{\partial x^2}+\frac{\partial^2(u)}{\partial y^2}=-|sin^3(\pi xy)| \tag{2}$$

$$x\in[-1,1], y\in[-1,1]$$

$$u(a,y)=\psi_1(y)=1-y^2,\quad u(b,y)=\psi_2(y)=1-y^2$$

$$u(x,c)=\psi_3(x)=|sin(\pi x)|,\quad u(x,d)=\psi_4(x)=|sin(\pi x)|$$

Абгрунтаванне метада

Апраксімуем другія вытворныя ў (1) з дапамогай рознасных схемаў. Атрымаем:

$$(y_{\overline{x}})_x+(y_{\overline{y}})_y=-f \tag{3}$$

$$y_{0j}=\psi_{1j},\quad y_{N_x,j}=\psi_{2j},\quad j=\overline{0,N_y}$$

$$y_{i0}=\psi_{3i},\quad y_{i,N_y}=\psi_{4i},\quad i=\overline{0,N_x}$$

Распішам у індэкснай форме:

$$\frac{y_{i+1,j}-2y_{ij}+y_{i-1,j}}{h_x^2}+\frac{y_{i,j+1}-2y_{ij}+y_{i,j-1}}{h_y^2}=-f_{ij},\ \ i=\overline{1,N_x-1}, j=\overline{1,N_y-1} \tag{4}$$

Знойдзем хібнасць такой рознаснай схемы - раскладзем $\psi=(u_{\overline{x}})_x+(u_{\overline{y}})_y+f$:

$$\psi=\frac{\partial^2u(x_i,y_j)}{\partial x^2}+\frac{h_x^2}{12}\frac{\partial^4u(x_i,y_j)}{\partial x^4}+\mathcal{O}(h_x^4)+\frac{\partial^2u(x_i,y_j)}{\partial y^2}+\frac{h_y^2}{12}\frac{\partial^4u(x_i,y_j)}{\partial y^4}+\mathcal{O}(h_y^4)+f_{ij}=\mathcal{O}(h_x^2+h_y^2) \tag{5}$$

Выразім з формулы (4) y_{ij}:

$$y_{ij}=(\frac{2}{h_x^2}+\frac{2}{h_y^2})^{-1}(f_{ij}+\frac{y_{i+1,j}+y_{i-1,j}}{h_x^2}+\frac{y_{i,j+1}+y_{i,j-1}}{h_y^2}),\ \ i=\overline{1,N_x-1}, j=\overline{1,N_y-1} \tag{6}$$

Формула (6) можа быць падлічаная любым з ітэратыўных метадаў рашэння нелінейных ураўненняў - па ўмовах маёй задачы, будзем прымяняць метад Зэйдэля:

$$y_{ij}^{k+1}=(\frac{2}{h_x^2}+\frac{2}{h_y^2})^{-1}(f_{ij}+\frac{y_{i+1,j}^{k}+y_{i-1,j}^{k+1}}{h_x^2}+\frac{y_{i,j+1}^{k}+y_{i,j-1}^{k+1}}{h_y^2}),\ \ i=\overline{1,N_x-1}, j=\overline{1,N_y-1} \tag{7}$$

Такім чынам, мы будзем рухацца па сетцы, з кута (a,c) у кут (b,d), першасна ўздоўж восі Ox, другасна - уздоўж Oy, паслядоўна для кожнага вузла выконваючы (7). Спыняемся, калі $\max|y_{ij}^{k+1}-y_{ij}^{k}|\le\epsilon, i=\overline{1,N_x-1}, j=\overline{1,N_y-1}$. Каб забяспечыць $\mathcal{O}(h_x^2+h_y^2)$ бярэм $\epsilon=\mathcal{O}(h^3)$ (каб дакладна забяспечыць $\epsilon=\mathcal{O}(h^2)$), $h=\min(h_x,h_y)$.

Рэалізацыя

```
# Defining the Dirichlet problem for Poisson equation.
f = lambda x, y: abs(math.sin(math.pi * x * y) ** 3)
psi_left = lambda y: 1 - y * y
psi_right = lambda y: 1 - y * y
psi_bottom = lambda x: abs(math.sin(math.pi * x))
psi_top = lambda x: abs(math.sin(math.pi * x))
a = -1.
b = 1.
c = -1.
d = 1.
# Algorithm parameters.
hx = 0.2
hy = 0.2
nx = int((b - a) / hx + 1)
ny = int((d - c) / hy + 1)
EPS = 1e-4
MAX_ZEIDEL_ITERATIONS = 1000

def dirichlet():
    grid = [[0. for x in range(nx)] for y in range(ny)]

    # Filling grid with border values.
    for i in range(0, nx):
        x = a + i * hx
        grid[0][i] = psi_bottom(x)
        grid[ny-1][i] = psi_top(x)
    for j in range(0, ny):
        y = c + j * hy
        grid[j][0] = psi_left(y)
        grid[j][nx-1] = psi_right(y)
    # Filling inner cells with f(i,j) - approximating.
    for i in range(1, nx-1):
        x = a + i * hx
        for j in range(1, ny-1):
            y = c + j * hy
            grid[i][j] = f(x, y)

    # Updating all inner cells.
    iter = MAX_ZEIDEL_ITERATIONS
    while iter > 0:
        mx_diff = 0
        for i in range(1, nx-1):
            x = a + i * hx
            for j in range(1, ny-1):
                y = c + j * hy
                oldValue = grid[i][j]
                grid[i][j] = 1. / (2/(hx*hx) + 2/(hy*hy)) * (f(x, y) + (grid[i-1][j] +
                        grid[i+1][j])/hx + (grid[i][j+1] + grid[i][j-1])/hy)
                mx_diff = max(mx_diff, abs(oldValue - grid[i][j]))
        if mx_diff < EPS:
            break
        iter -= 1
    if iter == 0:
        print "Zeidel: no convergence"
        return
    else:
        print "Number of iterations:", MAX_ZEIDEL_ITERATIONS - iter

    for i in range(nx-1, -1, -1):
        for j in range(0, ny):
            value = '{0:.7f} '.format(grid[i][j])
            sys.stdout.write(value)
        sys.stdout.write('\n')
    return
```

Вынік

Сеткі, прадстаўленыя ніжэй, арыентаваныя так, што ў левым ніжнім куце знаходзіцца кропка (a, c), у правым верхнім - (b, d).

Для сеткі з крокам $h_x = h_y = 0.2$:

```
0.0000000 0.5877853 0.9510565 0.9510565 0.5877853 0.0000000 0.5877853 0.9510565 0.9510565 0.5877853 0.0000000
0.3600000 0.0604217 0.0641429 0.0588235 0.0337566 0.0034073 0.0337526 0.0588207 0.0641423 0.0604205 0.3600000
0.6400000 0.0483340 0.0137298 0.0070617 0.0026013 0.0004726 0.0026170 0.0070696 0.0137290 0.0483243 0.6400000
0.8400000 0.0535519 0.0068643 0.0018596 0.0003962 0.0001096 0.0004614 0.0019084 0.0068635 0.0535175 0.8400000
0.9600000 0.0548107 0.0037883 0.0004563 0.0000689 0.0000338 0.0001263 0.0005278 0.0037978 0.0547691 0.9600000
1.0000000 0.0556680 0.0031904 0.0002132 0.0000197 0.0000096 0.0000403 0.0002532 0.0032100 0.0556459 1.0000000
0.9600000 0.0547803 0.0037716 0.0004527 0.0000646 0.0000124 0.0000673 0.0004609 0.0037818 0.0547733 0.9600000
0.8400000 0.0535216 0.0068545 0.0018577 0.0003865 0.0000694 0.0003895 0.0018574 0.0068544 0.0535201 0.8400000
0.6400000 0.0483285 0.0137286 0.0070606 0.0025909 0.0004533 0.0026078 0.0070697 0.0137282 0.0483246 0.6400000
0.3600000 0.0604334 0.0641468 0.0588249 0.0337636 0.0034294 0.0337945 0.0588554 0.0641515 0.0604213 0.3600000
0.0000000 0.5877853 0.9510565 0.9510565 0.5877853 0.0000000 0.5877853 0.9510565 0.9510565 0.5877853 0.0000000
```

Для сеткі з крокам $h_x = h_y = 0.1$:

```
0.0000 0.3090 0.5878 0.8090 0.9511 1.0000 0.9511 0.8090 0.5878 0.3090 0.0000 0.3090 0.5878 0.8090 0.9511 1.0000 0.9511 0.8090 0.5878 0.3090 0.0000
0.1900 0.0136 0.0168 0.0233 0.0276 0.0288 0.0270 0.0224 0.0159 0.0082 0.0004 0.0082 0.0159 0.0224 0.0270 0.0288 0.0276 0.0233 0.0168 0.0136 0.1900
0.3600 0.0109 0.0027 0.0032 0.0034 0.0031 0.0023 0.0015 0.0007 0.0003 0.0000 0.0003 0.0007 0.0015 0.0023 0.0031 0.0034 0.0032 0.0027 0.0109 0.3600
0.5100 0.0155 0.0030 0.0028 0.0025 0.0020 0.0013 0.0007 0.0002 0.0000 0.0000 0.0000 0.0002 0.0007 0.0013 0.0020 0.0025 0.0028 0.0030 0.0155 0.5100
0.6400 0.0195 0.0032 0.0025 0.0021 0.0015 0.0009 0.0004 0.0001 0.0000 0.0000 0.0000 0.0001 0.0004 0.0009 0.0015 0.0021 0.0025 0.0032 0.0195 0.6400
0.7500 0.0223 0.0029 0.0020 0.0015 0.0010 0.0006 0.0003 0.0001 0.0000 0.0000 0.0000 0.0001 0.0003 0.0006 0.0010 0.0015 0.0020 0.0029 0.0223 0.7500
0.8400 0.0241 0.0023 0.0013 0.0009 0.0006 0.0003 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0001 0.0003 0.0006 0.0009 0.0013 0.0023 0.0241 0.8400
0.9100 0.0251 0.0015 0.0007 0.0004 0.0003 0.0001 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0001 0.0001 0.0003 0.0004 0.0007 0.0015 0.0251 0.9100
0.9600 0.0257 0.0010 0.0002 0.0001 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0001 0.0001 0.0002 0.0010 0.0257 0.9600
0.9900 0.0261 0.0007 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0001 0.0007 0.0261 0.9900
1.0000 0.0263 0.0007 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0007 0.0263 1.0000
0.9900 0.0261 0.0007 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0001 0.0007 0.0261 0.9900
0.9600 0.0257 0.0010 0.0002 0.0001 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0001 0.0001 0.0002 0.0010 0.0257 0.9600
0.9100 0.0251 0.0015 0.0007 0.0004 0.0003 0.0001 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0001 0.0001 0.0003 0.0004 0.0007 0.0015 0.0251 0.9100
0.8400 0.0241 0.0023 0.0013 0.0009 0.0006 0.0003 0.0001 0.0000 0.0000 0.0000 0.0000 0.0000 0.0001 0.0003 0.0006 0.0009 0.0013 0.0022 0.0241 0.8400
0.7500 0.0223 0.0029 0.0020 0.0015 0.0010 0.0006 0.0003 0.0001 0.0000 0.0000 0.0000 0.0001 0.0003 0.0006 0.0010 0.0015 0.0020 0.0029 0.0223 0.7500
0.6400 0.0195 0.0032 0.0025 0.0021 0.0015 0.0009 0.0004 0.0001 0.0000 0.0000 0.0000 0.0001 0.0004 0.0009 0.0015 0.0021 0.0025 0.0032 0.0195 0.6400
0.5100 0.0155 0.0030 0.0028 0.0025 0.0020 0.0013 0.0007 0.0002 0.0000 0.0000 0.0000 0.0002 0.0007 0.0013 0.0020 0.0025 0.0028 0.0030 0.0155 0.5100
0.3600 0.0109 0.0027 0.0032 0.0034 0.0031 0.0023 0.0015 0.0007 0.0003 0.0000 0.0003 0.0007 0.0015 0.0023 0.0031 0.0034 0.0032 0.0027 0.0109 0.3600
0.1900 0.0136 0.0168 0.0234 0.0276 0.0289 0.0270 0.0224 0.0159 0.0082 0.0004 0.0082 0.0159 0.0224 0.0270 0.0289 0.0276 0.0234 0.0168 0.0136 0.1900
0.0000 0.3090 0.5878 0.8090 0.9511 1.0000 0.9511 0.8090 0.5878 0.3090 0.0000 0.3090 0.5878 0.8090 0.9511 1.0000 0.9511 0.8090 0.5878 0.3090 0.0000
```

Сетку, пабудаваную з крокамі $h_x = h_y = 0.05$ занадта складана размясціць праз вялікую колькасць вузлоў.
Метад будаваўся так, каб забяспечыць $\mathcal{O}(h^2)$, таму для вузлоў другой табліцы можам гарантаваць дакладнасць двух знакаў пасля коскі.

2 Рознасныя схемы для ўраўнення цеплаправоднасці

Пастаноўка задачы

Знайсці набліжанае рашэнне з дапамогай рознаснай схемы з вагамі σ на раўнамернай сетцы вузлоў $\overline{\omega}_{h\tau}$ з хібнасцю апраксімацыі не ніжэй за $\mathcal{O}(h^2 + \tau)$ для задачы выгляду:

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial^2 x} + f(x,t), \quad 0 \le x, t \le 1, \\ u(x,0) = u_0(x), \\ \alpha_0 u(0,t) + \beta_0 \frac{\partial u(0,t)}{\partial x} = \mu_0(t), \\ \alpha_1 u(1,t) + \beta_1 \frac{\partial u(1,t)}{\partial x} = \mu_1(t). \end{cases} \tag{8}$$

Мая канкрэтная задача выглядае наступным чынам:

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial^2 x} + x^2 - 2t, \quad 0 \le x, t \le 1, \\ u(x,0) = 0, \\ u(0,t) = 0, \\ \frac{\partial u(1,t)}{\partial x} = 2t, \\ \sigma = 0.5, \\ h = \tau = 0.05. \end{cases} \tag{9}$$

Абгрунтаванне метада

Для апраксімацыі будзем выкарыстоўваць рознасную схему з шасцікропкавым шаблонам:

$$y_t = \sigma y_{x\overline{x}} + (1 - \sigma) y_{\overline{x}x} + \varphi$$

Умова ўстойлівасці гэтай схемы: $\sigma \geq 0.5 - \frac{h^2}{4\tau}$. Калі σ адпавядае ўмове ўстойлівасці, а $\varphi \equiv f$, рознасная схема апраксімацыі ўраўнення будзе мець неабходны нам парадак апраксімацыі.

Разгледзім гранічныя ўмовы, пабудуем для іх рознасную схему і вызначым для гэтай схемы хібнасць апраксімацыі, разгледзеўшы нявязку на дакладным рашэнні.

Першая гранічная ўмова:

$$\begin{aligned}\alpha_0 u(x_0,t) + \beta_0 u(x_0,t)_x - \mu_0(t) = \alpha_0 u(x_0,t) + \frac{\beta_0}{h}(u(x_1,t) - u(x_0,t)) - \mu_0(t) = \\ = \alpha_0 u(x_0,t) + \frac{\beta_0}{h}(u(x_0,t) + h\frac{\partial u(x_0,t)}{\partial x} + \frac{h^2}{2}\frac{\partial^2 u(x_0,t)}{\partial^2 x} + \mathcal{O}(h^3) - u(x_0,t)) - \mu_0(t) = \\ = \alpha_0 u(x_0,t) + \beta_0\frac{\partial u(x_0,t)}{\partial x} + \frac{\beta_0 h}{2}\frac{\partial^2 u(x_0,t)}{\partial^2 x} + \mathcal{O}(h^2) - \mu_0(t) = \frac{\beta_0 h}{2}\frac{\partial^2 u(x_0,t)}{\partial^2 x} + \mathcal{O}(h^2) \quad (10)\end{aligned}$$

Другая гранічная ўмова:

$$\begin{aligned}\alpha_1 u(x_N,t) + \beta_1 u(x_N,t)_{\overline{x}} - \mu_1(t) = \alpha_1 u(x_N,t) + \frac{\beta_1}{h}(u(x_N,t) - u(x_{N-1},t)) - \mu_1(t) = \\ = \alpha_1 u(x_N,t) + \frac{\beta_1}{h}(u(x_N,t) - (u(x_N,t) - h\frac{\partial u(x_N,t)}{\partial x} + \frac{h^2}{2}\frac{\partial^2 u(x_N,t)}{\partial^2 x} + \mathcal{O}(h^3))) - \mu_1(t) = \\ = \alpha_1 u(x_N,t) + \beta_1\frac{\partial u(x_N,t)}{\partial x} - \frac{\beta_1 h}{2}\frac{\partial^2 u(x_N,t)}{\partial^2 x} + \mathcal{O}(h^2) - \mu_1(t) = -\frac{\beta_1 h}{2}\frac{\partial^2 u(x_N,t)}{\partial^2 x} + \mathcal{O}(h^2) \quad (11)\end{aligned}$$

Атрымліваецца парадак апраксімацыі $\mathcal{O}(h)$. Каб павысіць парадак да неабходнага, трэба ўвесці ў гранічныя ўмовы галоўны член хібнасці. Для гэтага выразім $\frac{\partial^2 u}{\partial^2 x}$ з зыходнага ўраўнення: $\frac{\partial^2 u}{\partial^2 x} = \frac{\partial u}{\partial t} - f(x,t)$, а першую прыватную вытворную па t будзем апраксімаваць пры дапамозе левай рознаснай вытворнай, то бок $\frac{\partial u}{\partial t} \approx u(x,t)_{\overline{t}}$.

Запішам атрыманыя ўмовы ў індэкснай форме:

$$\begin{cases} -\frac{\sigma}{h^2}y_{i-1,j+1} + (\frac{1}{\tau} + \frac{2\sigma}{h^2})y_{i,j+1} - \frac{\sigma}{h^2}y_{i+1,j+1} = \frac{y_{ij}}{\tau} + \frac{1-\sigma}{h^2}(y_{i+1,j} - 2y_{ij} + y_{i+1,j}) + f_{ij}, \quad i = \overline{1, N-1} \\ (\frac{\beta_0}{h} - \alpha_0 + \frac{\beta_0 h}{2\tau})y_{0,j+1} - \frac{\beta_0}{h}y_{1,j+1} = -\mu_{0,j+1} + \frac{\beta_0 h}{2\tau}(y_{0j} + \tau f_{0,j+1}), \\ -\frac{\beta_1}{h}y_{N-1,j+1} + (\alpha_1 + \frac{\beta_1}{h} + \frac{\beta_1 h}{2\tau})y_{N,j+1} = \mu_{N,j+1} + \frac{\beta_1 h}{2\tau}(y_{Nj} + \tau f_{N,j+1}), \\ j = \overline{0, N-1}. \end{cases} \quad (12)$$

$y_{i0}, i = \overline{0, N}$ вядомыя з умоваў. Каб атрымаць y_{i1}, трэба рашыць сістэму лінейных алгебраічных ураўненняў (12) адносна $y_{i,j+1}$, матрыца якой мае трохдыяганальны выгляд. Сістэму рашаем метадам прагонкі, па аналагічнай схеме з y_{i1} знаходзім y_{i2}, і гэтак далей.

Рэалізацыя

```
# Defining the problem.
f = lambda x, t: x * x - 2 * t
x_range = [0., 1.]
t_range = [0., 1.]
u0 = lambda x: 0.
m0 = lambda t: 0.
m1 = lambda t: 2 * t
a0 = 0
a1 = 0
b0 = 0
b1 = 1
s = 0.5
hx = 0.05
ht = 0.05

nx = int((x_range[1] - x_range[0]) / hx) + 1
nt = int((t_range[1] - t_range[0]) / ht) + 1
```

```
def solve_heat_equation():
    grid = [[0. for x in range(nt)] for y in range(nx)]
    for i in range(nx):
        x = x_range[0] + i * hx
        grid[0][i] = u0(x)

    for j in range(1, nt):
        t_temp = t_range[0] + j * ht
        # Running tridiagonal algorithm for each row:
        # Allocating memory.
        matr = [[0. for x in range(nx)] for y in range(nx)]
        rhs = [0. for x in range(nx)]
        for i in range(1, nx-1):
            matr[i][i-1] = -s/(hx*hx)
            matr[i][i] = (1./ht + 2.*s/(hx*hx))
            matr[i][i+1] = -s/(hx*hx)
            x_temp = x_range[0] + i * hx
            rhs[i] = grid[j-1][i] / ht + (1 - s) / (hx * hx) * (grid[j-1][i-1] -
                    2 * grid[j-1][i] + grid[j-1][i+1]) + f(x_temp, t_temp)

        matr[0][0] = b0/hx - a0 + b0*hx/(2*ht)
        matr[0][1] = -b0/hx
        rhs[0] = -m0(t_temp) + b0*hx/(2*ht)*(grid[j-1][0] + ht * f(x_range[0], t_temp-ht))

        matr[nx-1][nx-2] = -b1/hx
        matr[nx-1][nx-1] = a1 + b1/hx + b1*hx/(2*ht)
        rhs[nx-1] = m1(t_temp) + b1*hx/(2*ht) * (grid[j-1][nx-1] + ht * f(x_range[1], t_temp-ht))

        x = tridiagonal_matrix_algo(matr, rhs)
        for i in range(0, len(x)):
            grid[i][j] = x[i]

    for i in range(nx-1, -1, -1):
        for j in range(0, nt):
            value = '{0:.3f} '.format(grid[i][j])
            sys.stdout.write(value)
        sys.stdout.write('\n')
```

Вынік

Для сеткі на квадраце 1×1 з $h = \tau = 0.05$ атрымліваем наступныя значэнні:

```
-0.0000 0.0047 0.0143 0.0288 0.0481 0.0723 0.1014 0.1354 0.1743 0.2180 0.2666 0.3201 0.3785 0.4418 0.5100 0.5831 0.6610 0.7438 0.8315 0.9241 1.0216
-0.0000 0.0046 0.0138 0.0276 0.0460 0.0691 0.0968 0.1291 0.1661 0.2077 0.2539 0.3048 0.3603 0.4205 0.4853 0.5547 0.6287 0.7074 0.7908 0.8787 0.9714
-0.0000 0.0044 0.0132 0.0264 0.0439 0.0658 0.0922 0.1228 0.1579 0.1974 0.2412 0.2895 0.3421 0.3991 0.4605 0.5263 0.5964 0.6710 0.7500 0.8333 0.9211
-0.0000 0.0043 0.0126 0.0252 0.0418 0.0626 0.0875 0.1165 0.1497 0.1870 0.2285 0.2741 0.3238 0.3777 0.4357 0.4978 0.5641 0.6345 0.7091 0.7878 0.8707
-0.0000 0.0041 0.0121 0.0239 0.0397 0.0593 0.0828 0.1102 0.1415 0.1766 0.2157 0.2586 0.3055 0.3562 0.4108 0.4693 0.5317 0.5980 0.6682 0.7423 0.8203
-0.0000 0.0039 0.0115 0.0227 0.0375 0.0560 0.0781 0.1038 0.1332 0.1662 0.2029 0.2432 0.2871 0.3347 0.3859 0.4408 0.4993 0.5615 0.6273 0.6967 0.7698
-0.0000 0.0038 0.0109 0.0215 0.0354 0.0527 0.0734 0.0974 0.1249 0.1558 0.1900 0.2276 0.2687 0.3131 0.3609 0.4122 0.4668 0.5248 0.5863 0.6511 0.7193
-0.0000 0.0036 0.0103 0.0202 0.0332 0.0493 0.0686 0.0910 0.1166 0.1453 0.1771 0.2121 0.2502 0.2915 0.3359 0.3835 0.4343 0.4881 0.5452 0.6054 0.6688
-0.0000 0.0034 0.0097 0.0189 0.0310 0.0460 0.0638 0.0846 0.1082 0.1347 0.1641 0.1965 0.2317 0.2698 0.3108 0.3548 0.4016 0.4514 0.5041 0.5596 0.6181
-0.0000 0.0033 0.0091 0.0176 0.0288 0.0426 0.0590 0.0780 0.0998 0.1241 0.1511 0.1808 0.2131 0.2481 0.2857 0.3260 0.3689 0.4145 0.4628 0.5138 0.5674
-0.0000 0.0030 0.0085 0.0163 0.0265 0.0391 0.0541 0.0715 0.0913 0.1134 0.1380 0.1650 0.1944 0.2262 0.2604 0.2971 0.3361 0.3776 0.4215 0.4678 0.5165
-0.0000 0.0029 0.0079 0.0150 0.0242 0.0356 0.0492 0.0649 0.0827 0.1027 0.1249 0.1492 0.1757 0.2043 0.2351 0.2681 0.3032 0.3406 0.3801 0.4217 0.4656
-0.0000 0.0026 0.0072 0.0136 0.0219 0.0321 0.0442 0.0582 0.0741 0.0919 0.1116 0.1332 0.1568 0.1823 0.2097 0.2390 0.2702 0.3034 0.3385 0.3755 0.4145
-0.0000 0.0024 0.0065 0.0122 0.0195 0.0285 0.0391 0.0514 0.0654 0.0810 0.0982 0.1172 0.1378 0.1601 0.1841 0.2097 0.2371 0.2661 0.2968 0.3292 0.3633
-0.0000 0.0022 0.0058 0.0107 0.0171 0.0249 0.0340 0.0446 0.0566 0.0700 0.0848 0.1010 0.1187 0.1378 0.1584 0.1803 0.2038 0.2286 0.2550 0.2827 0.3119
-0.0000 0.0020 0.0051 0.0093 0.0146 0.0211 0.0288 0.0376 0.0477 0.0588 0.0712 0.0847 0.0995 0.1154 0.1325 0.1508 0.1703 0.1910 0.2129 0.2361 0.2604
-0.0000 0.0017 0.0042 0.0077 0.0121 0.0174 0.0235 0.0306 0.0386 0.0476 0.0574 0.0683 0.0800 0.0928 0.1064 0.1211 0.1366 0.1532 0.1707 0.1892 0.2087
-0.0000 0.0014 0.0035 0.0061 0.0095 0.0134 0.0181 0.0234 0.0294 0.0361 0.0435 0.0516 0.0604 0.0699 0.0802 0.0911 0.1028 0.1152 0.1283 0.1421 0.1567
-0.0000 0.0010 0.0025 0.0044 0.0067 0.0094 0.0125 0.0161 0.0201 0.0245 0.0294 0.0348 0.0406 0.0469 0.0537 0.0609 0.0687 0.0769 0.0856 0.0948 0.1045
-0.0000 0.0008 0.0017 0.0027 0.0038 0.0051 0.0066 0.0083 0.0103 0.0125 0.0149 0.0175 0.0204 0.0236 0.0269 0.0306 0.0344 0.0385 0.0429 0.0475 0.0523
 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
```

Метад будаваўся з дакладнасцю ў $\mathcal{O}(h^2)$, таму для кожнага вузла маем дакладнасць прынамсі ў 2 знака пасля коскі.