БЕЛАРУСКІ ДЗЯРЖАЎНЫ УНІВЕРСІТЭТ

Факультэт прыкладной матэматыкі і інфарматыкі

Метады лікавага аналізу

Справаздача па лабараторнай працы
па тэме

Метады рашэння гранічнай задачы для АДУ-2

Богдана Уладзіслава
3 курс, 3 група

Выкладчык:
Буднік А. М.

Мінск, 2016

Дадзеная краевая задача для АДУ-2 наступнага выгляду:

$$\begin{cases} (k(x)u'(x))' - q(x)u(x) = -f(x), \quad a \le x \le b \\ k(0)u'(0) = \alpha_0 u(0) - \mu_0, \\ -k(1)u'(1) = \alpha_1 u(1) - \mu_1. \end{cases} \tag{1}$$

Ніжэй мы разгледзім метады Рытца і сетак для рашэння задачы.
Мая задача выглядае наступным чынам:

$$\begin{cases} (e^x u'(x))' - e^x u(x) = -e^{-x}, \quad 0 \le x \le 1 \\ u'(0) = u(0) \\ -eu'(1) = u(1) \end{cases} \tag{2}$$

1 Метад Рытца

Ідэя ў прадстаўленні рашэння ў наступным выглядзе:

$$u_n(x) = \varphi_0(x) + \sum_{i=1}^{n} a_i \varphi_i(x) \tag{3}$$

φ_i - нейкія лінейна незалежныя, непарыўныя на $[a, b]$ функцыі, якія маюць, напрыклад, наступны выгляд:

$$\begin{cases} \varphi_i = x^{i+1}(x-1)^2, \quad i = \overline{1, n} \\ \varphi_0 = c_1 + c_2 x \end{cases} \tag{4}$$

Каэфіцыэнты c_1, c_2 знаходзяцца з пачатковых умоваў.
Такім чынам задача сводзіцца да пошуку каэфіцыэнтаў a_i ураўнення (3). Для гэтага скалярна дамножым ураўненне (3) на φ_i:

$$\begin{cases} \sum_{j=1}^{n} (A\varphi_j, \varphi_i) a_j = (\varphi_i, f) - (A\varphi_0, \varphi_i), \quad i = \overline{1, n} \\ (\varphi_i, f) = \int_0^1 \varphi_i f dx \\ (A\varphi_j, \varphi_i) = \int_0^1 (k(x)\varphi_i' \varphi_j' + q(x)\varphi_i \varphi_j) dx \end{cases} \tag{5}$$

Інтэгралы могуць быць падлічаныя, напрыклад, праз формулу Сімпсана:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{N} \frac{h}{6}(f_{i-1} + 4f_{i-\frac{1}{2}} + f_i), \quad h = \frac{b-a}{N} \tag{6}$$

Склаўшы сістэму лінейных алгебраічных ураўненняў адносна $a_i, i = \overline{1, n}$, рашаем яе, напрыклад, метадам Гаўса. Атрымаўшы значэнні a_i, падстаўляем іх у формулу (3) і знаходзім функцыі $u_n(x)$. Пасля знаходзім значэнні $u_n(x)$ на вызначанай сетцы, што і з'ўляецца вынікам працы алгарытма.
У нашай рэалізацыі $n = 5$, $N = 100$ (для падліку інтэграла). Кропкі пасля будуюцца на 10-кропкавай сетцы.

```
n_ritz = 5
fi0 = lambda x: 0.240 * x + 0.480
fi0_dx = lambda x: 0.240

def ritz():
    # Defining fi and fi_dx.
    fi = lambda x, i: (x - x0)**(i + 1) * (x - x1)**2
    fi_dx = lambda x, i: (i + 1) * (x - x0)**i * (x - x1)**2 + (x - x0)**(i + 1) * 2 * (x - x1)

    # Allocating memory for system of linear equations.
    matr = [[0. for x in range(n_ritz)] for y in range(n_ritz)]
    rhs = [0. for x in range(n_ritz)]

    for i in range (1, n_ritz+1):
        for j in range(1, n_ritz+1):
            func1 = lambda x: k(x) * fi_dx(x, i) * fi_dx(x, j) + q(x) * fi(x, i) * fi(x, j)
            matr[i-1][j-1] = intergrate_simpson(x0, x1, func1)
        func2 = lambda x: fi(x, i) * f(x)
        func3 = lambda x: k(x) * fi_dx(x, i) * fi0_dx(x) + q(x) * fi(x, i) * fi0(x)
```

```
        rhs[i-1] = intergrate_simpson(x0, x1, func2) - intergrate_simpson(x0, x1, func3)
    a = alg.solve(matr, rhs)

    x = []
    for i in range(0, len(grid)):
        temp = fi0(grid[i])
        for j in range(0, n_ritz):
            temp += a[j] * fi(grid[i], j+1)
        x.append(temp)

    return x
```

Рэалізацыя метада Сімпсана для прыблізнага падліку вызначанага інтэграла:

```
n_integrate = 100

def intergrate_simpson(a, b, f):
    sum = 0
    h = (b - a) / n_integrate
    for i in range (1, n_integrate):
        sum += f(a + h*(i-1)) + 4 * f(a + h*(i-0.5)) + f(a + h * i)
    sum = sum * h / 6.
    return sum
```

Атрыманы вынік:

```
Ritz:
y(0.0) = 0.48
y(0.1) = 0.509830793921
y(0.2) = 0.537994026666
y(0.3) = 0.560468053889
y(0.4) = 0.58072852305
y(0.5) = 0.601307384812
y(0.6) = 0.622386606891
y(0.7) = 0.643673645461
y(0.8) = 0.666359730496
y(0.9) = 0.692516022787
y(1.0) = 0.719836701642
```

2 Метад сетак

Будуем сетку $\overline{\omega}_h = \{x_i| \quad x_i = ih, i = \overline{0, N}\}$, $h = \frac{1}{N}$. Возьмем $N = 10$.
Складзем рознасны аналаг для дыферэнцыяльнай задачы (1): раскрыем дужкі ва ўраўненні і заменім аператары дыферэнцавання рознаснымі.

$$k(x)u^{''}(x) + k'(x)u'(x) - q(x)u(x) = -f(x), \quad u' \approx u_{\overset{\circ}{x}}, \quad u'' \approx u_{\overline{x}x} \tag{7}$$

Атрымліваем:

$$k_{x^o}u_{x^o} + ku_{\overline{x}x} - qu \approx -f.y(x_i) = y_i \approx u_i \Rightarrow k_{\overset{\circ}{x}}y_{\overset{\circ}{x}} + ky_{\overline{x}x} - qy = -f \tag{8}$$

Паколькі ва ўраўненні прысутнічае другая вытворная, то мінімальны шаблон, якім мы можам абысціцся - трохкропкавы: $\{x_{i-1}, x_i, x_{i+1}\}$. Атрымліваем наступную роўнасць:

$$\frac{k_{i+1} - k_{i-1}}{2h}\frac{y_{i+1} - y_{i-1}}{2h} + k_i\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - q_iy_i = -f_i, i = \overline{1, N-1}. \tag{9}$$

Для апраксімацыі гранічных умоваў будзем браць двухкропкавы шаблон і правую рознасную вытворную для першай гранічнай умовы і левую рознасную вытворную для другой гранічнай умовы адпаведна. Такім чынам, маем:

$$k_0y_{x,0} = \alpha_0y_0 - \mu_0, -k_Ny_{\overline{x},N} = \alpha_1y_N - \mu_1$$

$$k_0\frac{y_1 - y_0}{h} = \alpha_0y_0 - \mu_0, -k_N\frac{y_N - y_{N-1}}{h} = \alpha_1y_N - \mu_1$$

Ацэнім хібнасць апраксімацыі ўсіх рознасных ураўненняў. Для гэтага ацэнім нявязку рознаснага ўраўнення на дакладным рашэнні.

$$\psi_{1,i} = \frac{k_{i+1}-k_{i-1}}{2h}\frac{u_{i+1}-u_{i-1}}{2h} + k_i\frac{u_{i+1}-2u_i+u_{i-1}}{h^2} - q_iu_i + f_i =$$
$$= \frac{1}{4h^2}(k_i+hk_i^{'}+\frac{h^2}{2}k_i^{''}+\mathcal{O}(h^3)-(k_i-hk_i^{'}+\frac{h^2}{2}k_i^{''}+\mathcal{O}(h^3)))(u_i+hu_i^{'}+\frac{h^2}{2}u_i^{''}+\mathcal{O}(h^3)-(u_i-hu_i^{'}+\frac{h^2}{2}u_i^{''}+\mathcal{O}(h^3)))+$$
$$+\frac{k_i}{h^2}(u_i+hu_i^{'}+\frac{h^2}{2}u''_i+\frac{h^3}{6}u_i^{(3)}+\mathcal{O}(h^4)-2u_i+u_i-hu_i^{'}+\frac{h^2}{2}u''_i-\frac{h^3}{6}u_i^{(3)}+\mathcal{O}(h^4)) - q_iu_i+f_i = ... =$$
$$= k_i^{'}u_i^{'}+\mathcal{O}(h^2)+k_iu_i^{''}+\mathcal{O}(h^2)-q_iu_i+f_i = \mathcal{O}(h^2) \quad (10)$$

$$\psi_2 = k_0\frac{u_1-u_0}{h} - \alpha_0u_0+\mu_0 = \frac{k_0}{h}(u_0+hu_0^{'}+\frac{h^2}{2}u_0^{''}+\mathcal{O}(h^3)-u_0)-\alpha_0u_0+\mu_0 =$$
$$= k_0u_0^{'}+\frac{k_0h}{2}u_0^{''}+\mathcal{O}(h^2)-\alpha_0u_0+\mu_0 = \frac{k_0h}{2}u_0^{''}+\mathcal{O}(h^2) \quad (11)$$

$$\psi_3 = -k_N\frac{u_N-u_{N-1}}{h} - \alpha_1u_N+\mu_1 = -\frac{k_N}{h}(u_N-(u_N-hu_N^{'}+\frac{h^2}{2}u_N^{''}+\mathcal{O}(h^3)))-\alpha_1u_N+\mu_1 =$$
$$= -k_Nu_N^{'}+\frac{k_Nh}{2}u_N^{''}+\mathcal{O}(h^2)-\alpha_1u_N+\mu_N = \frac{k_Nh}{2}u_N^{''}+\mathcal{O}(h^2) \quad (12)$$

Заўважым, што хібнасць гранічных умоваў мае парадак $\mathcal{O}(h)$. Зробім так, каб парадак гэтых умоваў быў $\mathcal{O}(h^2)$, для гэтага ўвядзем старэйшы каэфіцыэнт хібнасці ў гранічныя ўмовы. Выразім u'' з умовы: $u^{''} = \frac{1}{k}(-f+qu-k^{'}u^{'})$, а ў атрыманым выразе вытворныя першага парадка апраксімуем на мінімальным шаблоне з першым парадкам (для першай – правая рознасная вытворная, для другой – левая рознасная вытворная), у выніку атрымаем такія гранічныя ўмовы другога парадка:

$$(\frac{hq_0}{2}+\alpha_0+\frac{k_0+k_1}{2h})y_0 - \frac{k_0+k_1}{2h}y_1 = \mu_0+\frac{hf_0}{2}, \quad (13)$$

$$-\frac{k_N+k_{N-1}}{2h}y_{N-1}+(\frac{k_N+k_{N-1}}{2h}+\alpha_1+\frac{hq_N}{2})y_N = \mu_1+\frac{hf_N}{2}. \quad (14)$$

Гэтыя дзве ўмовы разам з умовай (9) утвараюць сістэму лінейных алгебраічных ураўненняў адносна $y_i,\ \ i=\overline{0,N}$ з трохдыяганальнай матрыцай сістэмы. Такую сістэму можна эфектыўна рашыць метадам прагонкі.

```
def grid_algorithm():
    matr = [[0. for x in range(n+1)] for y in range(n+1)]
    rhs = [0. for x in range(n+1)]
    for i in range(1, n):
        x = x0 + h * i
        x_prev = x - h
        x_next = x + h
        matr[i][i-1] = k(x) / (h * h) - (k(x_next) - k(x_prev)) / (4 * h * h)
        matr[i][i] = - 2 * k(x) / (h * h) - q(x)
        matr[i][i+1] = (k(x_next) - k(x_prev)) / (4 * h * h) + k(x) / (h * h)
        rhs[i] = - f(x)

    matr[0][0] = - k(x0) / h - a0 - q(x0)*h/2 - (k(x0 + h) - k(x0))/(2 * h)
    matr[0][1] = k(x0) / h + (k(x0 + h) - k(x0))/(2 * h)
    rhs[0] = - m0 - f(x0)*h/2

    matr[n][n-1] = + k(x1) / h - (k(x1) - k(x1 - h))/(2 * h)
    matr[n][n] = - k(x1) / h - a1 - q(x1)*h/2 + (k(x1) - k(x1 - h))/(2 * h)
    rhs[n] = - m1 - f(x1)*h/2

    x = tridiagonal_matrix_algo(matr, rhs)
    return x

def tridiagonal_matrix_algo(matr, rhs):
    a = [0. for x in range(n+1)]
    c = [0. for x in range(n+1)]
```

```
    b = [0. for x in range(n+1)]
    al = [0. for x in range(n+2)]
    be = [0. for x in range(n+2)]
    x = [0. for x in range(n+1)]

    for i in range(1, n):
        a[i] = -matr[i][i-1]
        b[i] = -matr[i][i+1]
        c[i] = matr[i][i]
    b[0] = -matr[0][1]
    c[0] = matr[0][0]
    a[n] = -matr[n][n-1]
    c[n] = matr[n][n]

    al[1] = b[0] / c[0]
    be[1] = rhs[0] / c[0]
    for i in range(1, n):
        al[i+1] = b[i] / (c[i] - al[i]*a[i])
    for i in range(1, n+1):
        be[i+1] = (rhs[i] + be[i]*a[i]) / (c[i] - al[i]*a[i])

    x[n] = be[n+1]
    for i in range(n-1, -1, -1):
        x[i] = al[i+1]*x[i+1] + be[i+1]

    return x
```

```
Grid algorithm:
y(0.0) = 0.481826972814
y(0.1) = 0.523241027864
y(0.2) = 0.55477193286
y(0.3) = 0.579641357672
y(0.4) = 0.600341552792
y(0.5) = 0.618793677619
y(0.6) = 0.636473802056
y(0.7) = 0.654513001266
y(0.8) = 0.673776717109
y(0.9) = 0.694927551783
y(1.0) = 0.718474844464
```

Высновы

Можна рашыць дыферэнцыяльнае ўраўненне і атрымаць дакладнае рашэнне:

$$u(x) = 1.141e^{-1.618x} + 0.339e^{0.618x} - e^{-2x} \tag{15}$$

Ва ўраўненні (15) я замяніў дакладныя значэнні каэфіцыэнтаў на набліжэнні ў выглядзе дзесятковых дробаў для спрашчэння падлікаў - на канчатковы вынік падобная аперацыя будзе мець мінімальны ўплыў. Падлічым дакладныя значэнні на сетцы:

```
Exact solution:
y(0.0) = 0.48
y(0.1) = 0.512427856823
y(0.2) = 0.53883854757
y(0.3) = 0.561472157666
y(0.4) = 0.582062960216
y(0.5) = 0.601950592919
y(0.6) = 0.622168463031
y(0.7) = 0.643513918114
y(0.8) = 0.666603837732
y(0.9) = 0.691918587652
y(1.0) = 0.719836701642
```

Знойдзем хібнасці абодвух метадаў, то бок для кожнага x_i з сеткі знойдзем розніцу $u_i - y_i = u(x_i) - y(x_i)$, дзе u_i - дакладнае значэнне, y_i - набліжэнне.

Метад Рытца:

```
Ritz:
u(0.0) - y(0.0) = 0.0
u(0.1) - y(0.1) = 0.00259706290159
u(0.2) - y(0.2) = 0.000844520904628
u(0.3) - y(0.3) = 0.00100410377728
u(0.4) - y(0.4) = 0.00133443716583
u(0.5) - y(0.5) = 0.0006432081075
u(0.6) - y(0.6) = -0.000218143860458
u(0.7) - y(0.7) = -0.000159727346558
u(0.8) - y(0.8) = 0.000244107235884
u(0.9) - y(0.9) = -0.000597435135413
u(1.0) - y(1.0) = 0.0
```

Метад сетак $O(h^2)$:

```
Grids:
u(0.0) - y(0.0) = -0.00182697281369
u(0.1) - y(0.1) = -0.010813171041
u(0.2) - y(0.2) = -0.0159333852899
u(0.3) - y(0.3) = -0.0181692000061
u(0.4) - y(0.4) = -0.0182785925757
u(0.5) - y(0.5) = -0.0168430846996
u(0.6) - y(0.6) = -0.0143053390252
u(0.7) - y(0.7) = -0.0109990831515
u(0.8) - y(0.8) = -0.00717287937789
u(0.9) - y(0.9) = -0.00300896413092
u(1.0) - y(1.0) = 0.00136185717777
```

Метад Рытца ў розных вяршынях сеткі дае хібнасць у межах $0.01 - 0.001$, метад сетак атрымаўся менш дакладным, але заяўленая дакладнасць (2-гі парадак) дасягаецца. Кожны з гэтых алгарытмаў патрабуе рэалізацыі дадатковых функцый - падлік інтэграла альбо рэалізацыя метада прагонкі. Кожны з іх можа быць паспяхова прыменены на практыцы.