Федеральное государственное автономное образовательное учреждение высшего образования «Московский физико-технический институт (национальный исследовательский университет)»

Физтех-школа аэрокосмических технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ – 3

Поиск численного решения краевой задачи

Выполнил:
Алиев Артем Эльдарович

Группа Б03-907

Долгопрудный

2021 г.

Содержание

1 Условие……**3**

2 Теория……**3**

3 Решение……**5**

3.1 Поиск матрицы ……5

3.2 Поиск решения ЗК методом непосредственной аппроксимации дифференциального уравнения с прогонкой……6

4 Результаты……**7**

5 Итог……**12**

1 Условие

Найти решение задачи Коши: *y'' – 2y' = exp(x)*(x² – 1), y(0) = 1, y(b) = -2, b = 1.5; 3.0, h = 0.02; 0.01; 0.005*, используя метод непосредственной аппроксимации дифференциального уравнения с прогонкой.

2 Теория

Непосредственная разностная аппроксимация их краевой задачи

$$y'' \longrightarrow \frac{y_{k+1} - 2y_k + y_{k-1}}{h^2}$$

$$y' \longrightarrow \frac{y_{k+1} - y_{k-1}}{2h}$$

$$p(x) \longrightarrow p_k, f(x) \longrightarrow f_k, q(x) \longrightarrow q_k$$

Далее с помощью метода трехдиагональной прогонки и домножения главного уравнения на h^2 задача решается.

Метод прогонки

Рассмотрим линейную задачу

$$\frac{d}{dx}\left[g(x)\frac{dy}{dx}\right] + q(x)\frac{dy}{dx} - p(x)y = f, \quad x \in [0,1],$$

с граничными условиями третьего рода (2.5).

Коэффициент $g(x)$, вообще говоря, может не иметь первой производной. Такая задача возникает, например, в случае расчета установившегося распределения температуры в задаче стационарной теплопроводности с разрывом коэффициента теплопроводности. Чтобы получить разностную схему, пригодную и для этого случая, представим разностную задачу в виде

$$\frac{1}{h}\left(g_{n+1/2}\frac{y_{n+1} - y_n}{h} - g_{n-1/2}\frac{y_n - y_{n-1}}{h}\right) + q_n\frac{y_{n+1} - y_{n-1}}{2h} - p_n y_n = f_n,$$

$$n = 1, \ldots, N-1,$$

$$g_{n+1/2} = g\left(x_n + 0{,}5h\right), \; q_n = q(x_n),$$

$$A_1\frac{y_1 - y_0}{h} + B_1 y_0 = U_1, \; x = 0,$$

$$A_2\frac{y_N - y_{N-1}}{h} + B_2 y_N = U_2, \; x = 1.$$

Здесь для аппроксимации производных в граничных условиях использована формула первого порядка аппроксимации. О недостатках такого подхода и способах их преодоления смотри ниже.

Для определения значений сеточной функции получается СЛАУ с трехдиагональной матрицей:

$$-b_0 y_0 + c_0 y_1 = d_0,$$

$$a_n y_{n-1} - b_n y_n + c_n y_{n+1} = d_n, \quad n = 1, \ldots, N-1,$$

$$a_N y_{N-1} - b_N y_N = d_N,$$

где введены обозначения $a_n = g_{n-1/2} - \frac{q_n h}{2}$, $c_n = g_{n+1/2} + \frac{q_n h}{2}$,

$b_n = a_n + c_n + h^2 p_n$, $d_n = h^2 f_n$, $b_0 = \frac{A_1}{h} - B_1$, $c_0 = \frac{A_1}{h}$, $d_0 = U_1$,

$b_N = \frac{A_2}{h} + B_2$, $a_N = -\frac{A_2}{h}$, $d_N = -U_2$.

Эта СЛАУ записывается в матричном виде $\mathbf{A'y} = \mathbf{d}$, где $\mathbf{A'}$ — матрица специального вида (ленточная, трехдиагональная):

$$\mathbf{A'} = \begin{pmatrix} -b_0 & c_0 & 0 & \ldots & 0 \\ a_1 & -b_1 & c_1 & \ldots & 0 \\ 0 & a_2 & -b_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & -b_N \end{pmatrix},$$

$\mathbf{y}$, $\mathbf{d}$ есть векторы-столбцы искомого сеточного решения и преобразованной правой части:

$$\mathbf{y} = (y_0, y_1, \ldots, y_N)^{\mathrm{T}}, \quad \mathbf{d} = (d_0, d_1, \ldots, d_N)^{\mathrm{T}}.$$

Трехдиагональные матрицы часто возникают при численном решении краевых задач как для обыкновенных дифференциальных уравнений, так и для уравнений в частных производных.

Рассмотрим экономичный вариант метода Гаусса, предназначенный для решения подобных систем. Решение ищется в виде прогоночного соотношения

$$y_{n-1} = P_n y_n + Q_n, \quad n = 1, \ldots, N, \tag{4.1}$$

Левое краевое условие также записывается в виде прогоночного соотношения

$$y_0 = \frac{c_0}{b_0} y_1 - \frac{d_0}{b_0},$$

откуда сразу получаем $P_1 = c_0 / b_0$, $Q_1 = -d_0 / b_0$.

Получим рекуррентные формулы, позволяющие последовательно вычислить все прогоночные коэффициенты.

Подставив равенство $y_{n-1} = P_n y_n + Q_n$ в уравнение

$$a_n y_{n-1} - b_n y_n + c_n y_{n+1} = d_n,$$

получим

$$a_n (P_n y_n + Q_n) - b_n y_n + c_n y_{n+1} = d_n,$$

или

$$y_n = \frac{c_n}{b_n - a_n P_n} y_{n+1} + \frac{a_n Q_n - d_n}{b_n - a_n P_n}.$$

Сравнивая эту запись с видом прогоночного соотношения (4.1), мы видим, что для прогоночных коэффициентов должны выполняться равенства

$$P_{n+1} = \frac{c_n}{b_n - a_n P_n}, \quad Q_{n+1} = \frac{a_n Q_n - d_n}{b_n - a_n P_n}.$$

Эти формулы определяют прямой ход прогонки, при этом последовательно слева направо определяются коэффициенты P_n и Q_n.

Из краевого условия на правом конце отрезка интегрирования

$$a_N y_{N-1} - b_N y_N = d_N$$

и прогоночного соотношения

$$y_{N-1} = P_N y_N + Q_N$$

находим

$$Q_{N+1} = \frac{a_N Q_N - d_N}{b_N - a_N P_N} \text{ и } y_N = Q_{N+1}.$$

Далее последовательно справа налево вычисляются остальные неизвестные y_n $n = N-1, \ldots, 1$ с использованием прогоночных соотношений и найденных при прямом ходе прогоночных коэффициентов. Это — обратный ход алгоритма прогонки.

Теорема. *Пусть выполнены условия диагонального преобладания* $|b_n| \geq |a_n| + |c_n|$ *и хотя бы для одной строки матрицы системы имеет место строгое диагональное преобладание* ($|b_n| > |a_n| + |c_n|$). *Пусть, кроме того,* $0 < P_1 < 1$. *Тогда алгоритм прогонки устойчив.*

3 Решение

3.1 Поиск матрицы

```
# Пробные данные для уравнения A*X = B
a = [[ 10.8000, 0.0475,      0, 0     ],
     [  0.0321, 9.9000, 0.0523, 0     ],
     [       0, 0.0369, 9.0000, 0.0570],
     [       0,      0, 0.0416, 8.1000]]

b = [12.1430, 13.0897, 13.6744, 13.8972]


def solution(a, b):

    n = len(a)
    x = [0 for k in range(0, n)] # обнуление вектора решений
    print('Размерность матрицы: ',n,'x',n)

    # Прямой ход
    v = [0 for k in range(0, n)]
    u = [0 for k in range(0, n)]
    # для первой 0-й строки
    v[0] = a[0][1] / (-a[0][0])
    u[0] = ( - b[0]) / (-a[0][0])
    for i in range(1, n - 1): # заполняем за исключением 1-й и (n-1)-
й строк матрицы
        v[i] = a[i][i+1] / ( -a[i][i] - a[i][i-1]*v[i-1] )
        u[i] = ( a[i][i-1]*u[i-1] - b[i] ) / ( -a[i][i] - a[i][i-1]*v[i-
1] )
    # для последней (n-1)-й строки
    v[n-1] = 0
    u[n-1] = (a[n-1][n-2]*u[n-2] - b[n-1]) / (-a[n-1][n-1] - a[n-1][n-
2]*v[n-2])


Размерность матрицы:  4 x 4
Решение:
x[0] =   1.1186
x[1] =   1.3106
x[2] =   1.5032
x[3] =   1.7080
```

3.2 Поиск решения ЗК методом непосредственной аппроксимации дифференциального уравнения с прогонкой

```
for b_el in b:
    y_arrays = []
    x_arrays = []
    length = b_el - a
    for h_el in h:
        N = int(length / h_el)
        # Create matrix of zeros of given size (remember about already giv
en 2 dots)
        A = np.zeros([N + 1, N + 1])

        # Create respective vector of x (remember about already given 2 do
ts) with a step of h
        x = np.linspace(a, b_el, num=N+1, endpoint=True)

        # Create vector of free coefficients

        d = [y_0] + [pow(h_el, 2) * np.exp(x[k]) * (x[k]**2 - 1) for k in
range(1, len(x)-1)] + [y_b]

        '''
        Straight move
        '''

        # Computing A matrix with 3 diagonals
        # e.g.
        # 1 0 0 0 0
        # 1 2 1 0 0
        # 0 1 5 1 0
        # 0 0 1 8 1
        # 0 0 0 0 1

        for i in range(N + 1):
            for j in range(N + 1):
                if i == j == 0 or i == j == N:
                    A[i][j] = 1
                elif i == j != 0 or i == j != N:
                    A[i][j] = 1 + h_el
                    A[i][j - 1] = 1 - h_el
                    A[i][j + 1] = 5 * pow(h_el, 2) - 2
```

4 Результаты

Выбрал 17 значений с шагом кратному h = 0.02 при b = 1.5:

Method: b = 1.5, h = 0.02		Exact value	Error
x	y	y	|Method - Exact value|
0,00	1,000000000	0,999996000	0,000004000
0,08	-1,738427225	1,005829084	2,744256309
0,18	-1,916302201	1,003873892	2,920176093
0,28	-1,917058741	0,988694597	2,905753338
0,38	-1,922812367	0,956691296	2,879503662
0,48	-1,928433369	0,903790659	2,832224029
0,58	-1,934097500	0,825450644	2,759548144
0,68	-1,939839509	0,716689270	2,656528778
0,78	-1,945709089	0,572146826	2,517855915
0,88	-1,951768633	0,386193528	2,337962162
0,98	-1,958095815	0,153098056	2,111193871
1,08	-1,964786465	-0,132723368	1,832063098
1,18	-1,971957923	-0,476352392	1,495605531
1,28	-1,979752926	-0,881876887	1,097876038
1,38	-1,988344131	-1,351722429	0,636621703
1,48	-1,997939358	-1,885756908	0,112182449
1,50	-2,000000000	-2,000007435	0,000007435
Max Error		4,326596344	

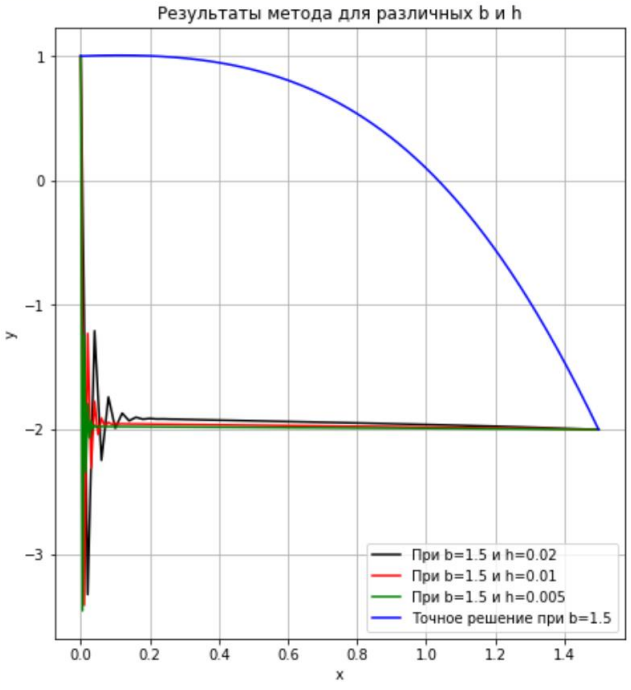
Выбрал 17 значений с шагом кратному h = 0.01 при b = 1.5:

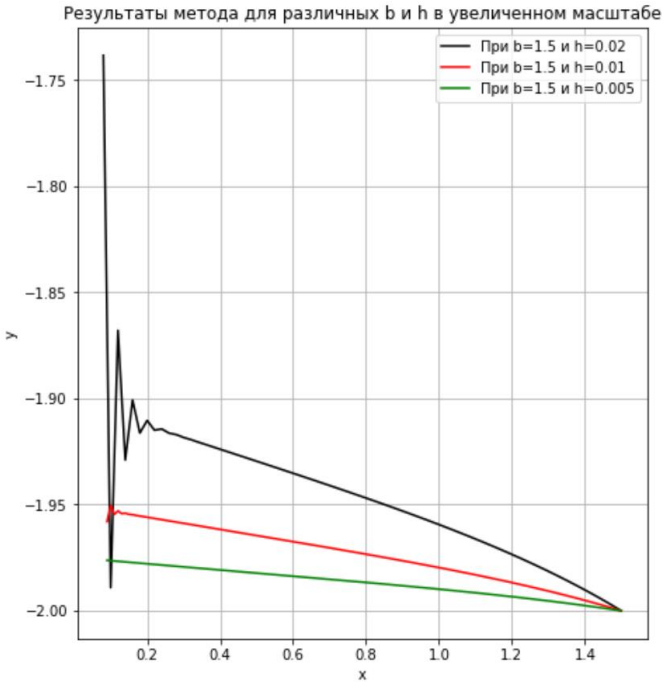
Method: b = 1.5, h = 0.01		Exact value	Error
x	y	y	|Method - Exact value|
0,00	1,000000000	0,999996000	0,000004000
0,09	-1,958035380	1,006135676	2,964171056
0,19	-1,955664279	1,003006769	2,958671048
0,29	-1,958532764	0,986314831	2,944847595
0,39	-1,961395755	0,952412933	2,913808688
0,49	-1,964260261	0,897180329	2,861440590
0,59	-1,967141656	0,816029215	2,783170871
0,69	-1,970059826	0,703936394	2,673996220
0,79	-1,973040064	0,555509432	2,528549496
0,89	-1,976114135	0,365099647	2,341213782
0,99	-1,979321511	0,126977739	2,106299250
1,09	-1,982710819	-0,164407802	1,818303018
1,19	-1,986341522	-0,514063948	1,472277574
1,29	-1,990285869	-0,925945738	1,064340131
1,39	-1,994631167	-1,402267510	0,592363657
1,49	-1,999482407	-1,942581616	0,056900791
1,50	-2,000000000	-2,000007435	0,000007435
Max Error		4,411906874	

Выбрал 17 значений с шагом кратному h = 0.005 при b = 1.5:

Method: b = 1.5, h = 0.005		Exact value	Error
x	y	y	|Method - Exact value|
0,000	1,000000000	0,999996000	0,000004000
0,095	-1,976342798	1,006250371	2,982593170
0,195	-1,977799299	1,002522703	2,980322002
0,295	-1,979250643	0,985060801	2,964311444
0,395	-1,980696040	0,950194180	2,930890219
0,495	-1,982141402	0,893778407	2,875919808
0,595	-1,983594523	0,811202912	2,794797435
0,695	-1,985065460	0,697424132	2,682489592
0,795	-1,986566989	0,547033686	2,533600675
0,895	-1,988115138	0,354374093	2,342489231
0,995	-1,989729806	0,113718016	2,103447822
1,095	-1,991435490	-0,180468563	1,810966927
1,195	-1,993262121	-0,533153398	1,460108724
1,295	-1,995246052	-0,948222555	1,047023497
1,395	-1,997431186	-1,427781337	0,569649848
1,495	-1,999870293	-1,971219804	0,028650489
1,500	-2,000000000	-2,000007435	0,000007435
Max Error		4,455594449	

Графически представленные результаты при b = 1.5:





Выбрал 17 значений с шагом кратному h = 0.02 при b = 3.0:

Method: b = 3.0, h = 0.02		Exact value	Error
x	y	y	|Method - Exact value|
0,00	1,00000000	1,00000000	0,00
0,18	-1,33401212	0,97596368	0,18
0,38	-1,33754833	0,88336982	0,38
0,58	-1,34489086	0,68438352	0,58
0,78	-1,35253253	0,33001508	0,78
0,98	-1,36092258	-0,23980439	0,98
1,18	-1,37076109	-1,09417830	1,18
1,38	-1,38309659	-2,30509471	1,38
1,58	-1,39946210	-3,93405423	1,58
1,78	-1,42205787	-6,00647067	1,78
1,98	-1,45399554	-8,46591536	1,98
2,18	-1,49962312	-11,09577419	2,18
2,38	-1,56495561	-13,38899157	2,38
2,58	-1,65824350	-14,33605819	2,58
2,78	-1,79072109	-12,08541324	2,78
2,98	-1,97758824	-3,40620232	2,98
3,00	-2,00000000	-2,00008118	3,00
Max Error		12,698685625	

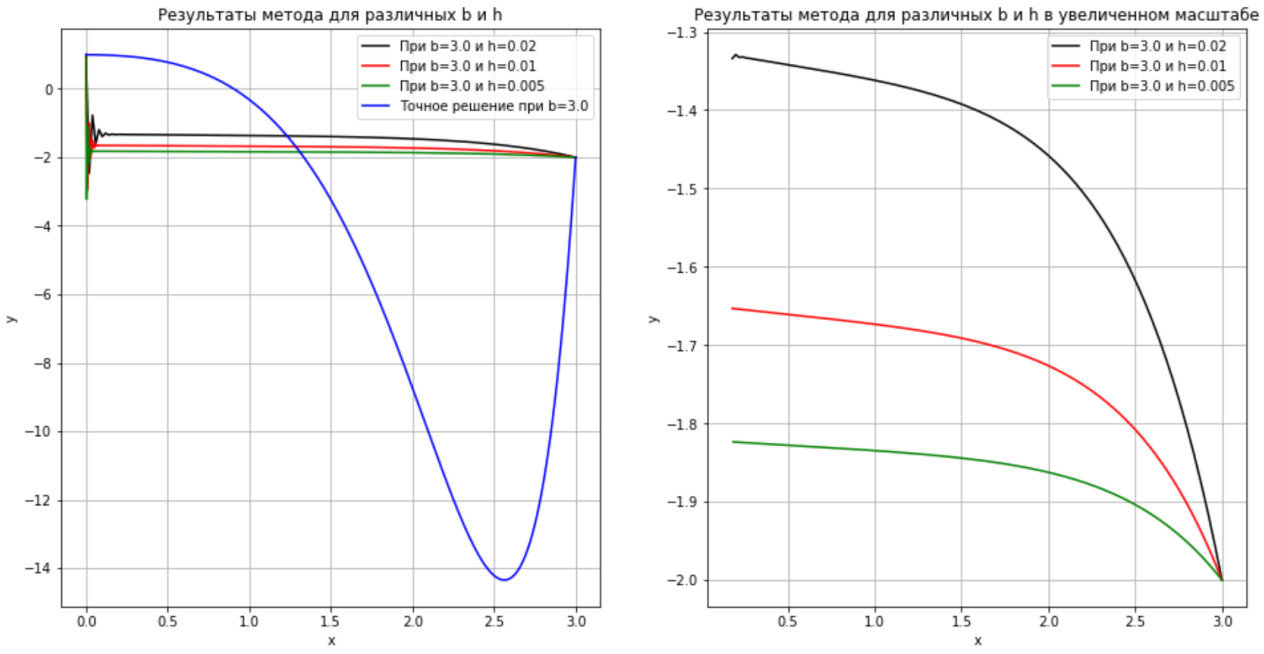
Выбрал 17 значений с шагом кратному h = 0.01 при b = 3.0:

Method: b = 3.0, h = 0.01		Exact value	Error
x	y	y	|Method - Exact value|
0,00	1,0000000	1,0000000	0,000000000
0,19	-1,6534488	0,9732313	2,626680121
0,39	-1,6581682	0,8763089	2,534477020
0,59	-1,6628981	0,6708109	2,333709027
0,79	-1,6677771	0,3071849	1,974962004
0,99	-1,6730358	-0,2751633	1,397872486
1,19	-1,6790296	-1,1456722	0,533357369
1,39	-1,6862895	-2,3762006	0,689911069
1,59	-1,6955906	-4,0271823	2,331591698
1,79	-1,7080429	-6,1212369	4,413194026
1,99	-1,7252145	-8,5961264	6,870911952
2,19	-1,7492939	-11,2243456	9,475051708
2,39	-1,7833058	-13,4795940	11,696288196
2,59	-1,8313954	-14,3196352	12,488239832
2,79	-1,8992028	-11,8387268	9,939524020
2,99	-1,9943539	-2,7180665	0,723712605
3,00	-2,0000000	-2,0000812	0,000081177
Max Error		12,522967080	

Выбрал 17 значений с шагом кратному h = 0.005 при b = 3.0:

Method: b = 3.0, h = 0.005		Exact value	Error
x	y	y	|Method - Exact value|
0,000	1,0000000	1,0000000	0,000000000
0,195	-1,8237475	0,9718006	2,795548096
0,395	-1,8263869	0,8726778	2,699064666
0,595	-1,8290275	0,6638777	2,492905240
0,795	-1,8317417	0,2955660	2,127307713
0,995	-1,8346458	-0,2931120	1,541533807
1,195	-1,8379190	-1,1717569	0,666162179
1,395	-1,8418285	-2,4121501	0,570321540
1,595	-1,8467639	-4,0741680	2,227404077
1,795	-1,8532828	-6,1789903	4,325707511
1,995	-1,8621724	-8,6614008	6,799228373
2,195	-1,8745307	-11,2883262	9,413795513
2,395	-1,8918744	-13,5236444	11,631770067
2,595	-1,9162813	-14,3084365	12,392155181
2,795	-1,9505776	-11,7093640	9,758786461
2,995	-1,9985831	-2,3628598	0,364276764
3,000	-2,0000000	-2,0000812	0,000081177
Max Error		12,434662232	

Графически представленные результаты при b = 1.5:



Точное решение:

$$y(x) = \frac{1}{2} c_1 e^{2x} + c_2 - e^x x^2 - e^x$$

Итоговая таблица со всеми погрешностями при различных b и h:

Max Error					
b = 1.5			b = 3.0		
h = 0.02	h = 0.01	h = 0.005	h = 0.02	h = 0.01	h = 0.005
4,32659634	**4,411907**	**4,455594**	**12,69869**	**12,52297**	**12,43466**

5 Итог

В данной задаче мы воспользовались методом непосредственной аппроксимации дифференциального уравнения с прогонкой. Для b = 3.0 от уменьшения h погрешность уменьшается, хоть и по-прежнему остается достаточно большой, как мы видим графически точное решение и наш метод значительно отличаются, однако для b = 1.5 от уменьшения h погрешность только возрастает, так как точное решение на графике находится выше, чем наше решение методом непосредственной аппроксимации дифференциального уравнения с прогонкой. Судя по всему данный метод не подходит для данной задачи.Ы

Ссылка на GitHub кода