САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Исследование нестационарного поля температур в плоской неограниченной пластине.

Группа : 23634/1

Студент ________________ Капицин Д. Р.

Преподаватель ________________ Плетнев А. А.

Санкт-Петербург
2018

Содержание

I. Физическая постановка задачи

Плоская неограниченная пластина из стекла толщиной $10cm$ испытывает конвективный теплообмен с оружающей средой (с обеих сторон пластины интенсивность конвективного теплообмена одинакова). В начальный момент времени температура пластины постоянна во всем сечении и равна $20\,C^{\circ}$. Температура окружающей среды $100\,C^{\circ}$. Найти распределение температуры пластины в зависимости от координаты и времени для трех значений коэффициента конвективной теплоотдачи:

$\alpha_1 = 7.4$ *Вт/(м^2 К)*; $\alpha_2 = 1500$ *Вт/(м^2 К)*; $\alpha_3 = 11000$ *Вт/(м^2 К)*.

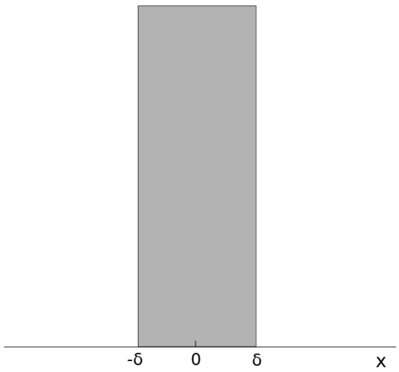


Рис. 1.1. Схема рассчетной области

1.1. Физические свойства кварцевого стекла (прозрачное стекло)

Основы теплопередачи М. А. Михеев И. М. Михеева — с. 316

Плотность [*кг/м^3*]	2500
Удельая теплоемкость при $20^o\,C$ [*кДж/кг К*]	0.67
Коэффициет теплопроводности при $20^o\,C$ [*Вт/(м К)*]	0.74

II. Математическая постановка задачи

$$C\rho\frac{\partial T}{\partial \tau} = \lambda\frac{\partial^2 T}{\partial x^2} \tag{2.1}$$

где λ – коэффициент теплопроводности, C – удельная тепроемкость , ρ – плотность.

Граничны условия:

$$\begin{aligned}
&\tau = 0; && T(0,x) = T_0 \\
&x = 0; && \lambda\frac{\partial T}{\partial x} = 0 \\
&x = \delta; && -\lambda\frac{\partial T}{\partial x} = \alpha(T - T_e)
\end{aligned}$$

Переход к безразмерным параметрам

$$\Theta = \frac{T - T_e}{T_0 - T_e}, \quad \xi = \frac{x}{\delta}, \quad a = \frac{\lambda}{C\rho}, \quad F_0 = \frac{a\tau}{\delta^2}, \quad Bi = \frac{\alpha\delta}{\lambda} \tag{2.2}$$

где a – коэффициент температуропроводности, Bi – безразметный коэффициент теплоотдачи (число Био), F_0 – безразмерное время (число Фурье).

Тогда уравнение 2.1 перепишется в виде

$$\frac{\partial\Theta}{\partial Fo} = \frac{\partial^2\Theta}{\partial\xi^2} \tag{2.3}$$

Граничны условия примут вид:

$$\begin{aligned} Fo &= 0; & \Theta &= 1 \\ \xi &= 0; & \frac{\partial\Theta}{\partial\xi} &= 0 \\ \xi &= 1; & -\frac{\partial\Theta}{\partial\xi} &= Bi \end{aligned}$$

III. Метод разделения координат

$$\Theta(Fo, \xi) = \phi(Fo)\,\psi(\xi) \tag{3.1}$$

$$\frac{\phi'}{\phi} = -\mu^2 = const \tag{3.2}$$

$$\begin{cases} \phi' + \mu^2\phi = 0 \\ \psi'' + \mu^2\psi = 0 \end{cases} \tag{3.3}$$

$$\begin{cases} \phi(Fo) = \exp(-\mu^2 Fo) C_1 \\ \psi(\xi) = C_2 \cos(M\xi) + C_3 \sin(M\xi) \end{cases} \tag{3.4}$$

$$\Theta(Fo, \xi) = \sum_{n=1}^{\infty} \frac{2\mu_n}{\sin\mu_n\,\cos\mu_n} \cos(\mu_n\xi)\,\exp(-\mu_n Fo) \tag{3.5}$$

Метод разделения координат позволяет решить краевую задачу аналитически. Тогда вычисление искомой функции $\Theta(Fo, \psi)$ сводиться к вычислению суммы ряда 3.5

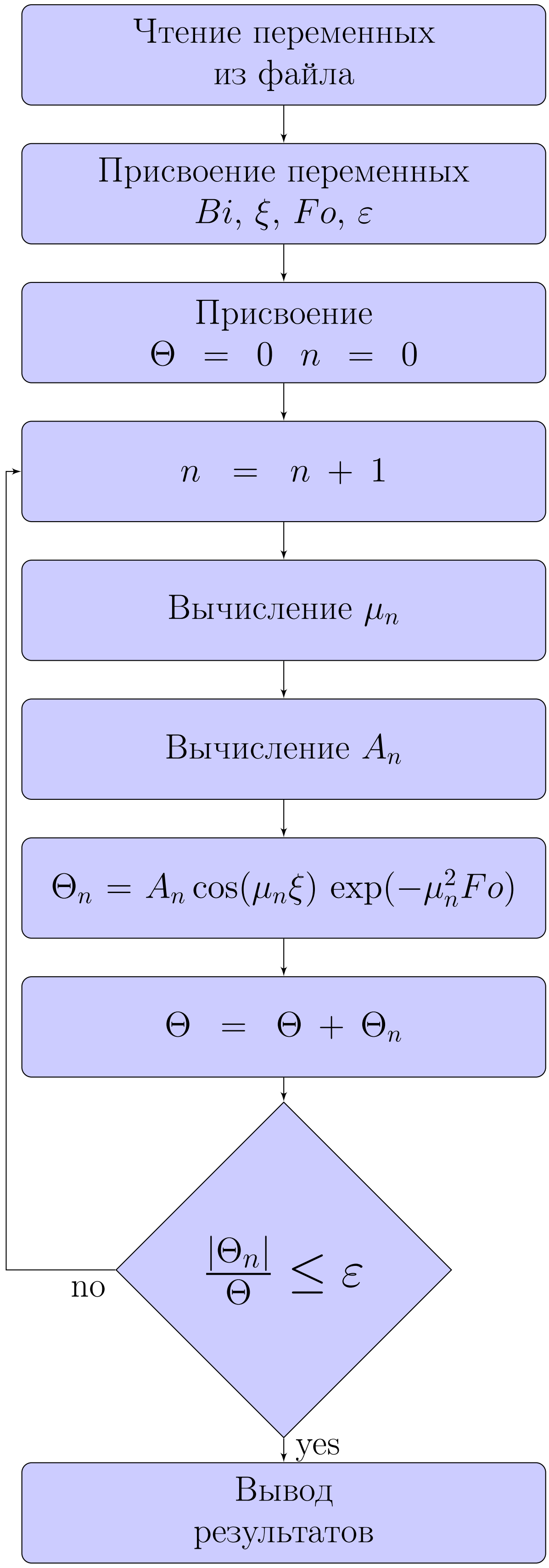


Рис. 3.1. Блок схема алгоритма разделения координат

IV. Метод конечных разностей

4.1. Явная схема

$$\frac{\partial T}{\partial x} = \frac{T_i^{n+1} - T_i^n}{\tau} \tag{4.1}$$

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{h^2} \tag{4.2}$$

В результате аппроксимации частных производных соответствующими конечными разностями получим следующую систему линейных алгебраических уравнений:

$$\rho C \frac{T_i^{n+1} - T_i^n}{\tau} = \lambda \left(\frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{h^2} \right), \qquad i = \overline{2, N-1} \tag{4.3}$$

или

$$T_i^{n+1} = T_i^n + \frac{\lambda \tau}{\rho C} \left(\frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{h^2} \right), \qquad i = \overline{2, N-1} \tag{4.4}$$

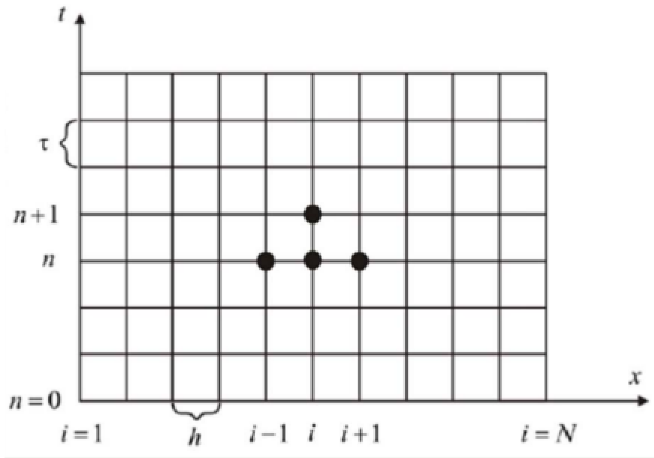


Рис. 4.1. Метод конечных разностей (явный)

Явная схема алгоритма конечных разностей неустойчива. Поэтому интегрирование краевой задачи необхоимо проводить с шагом по времени $\tau \le h^2/2$.

4.2. Неявная схема

Вслучае неявной схемы, уравнение (4.4) записывается в виде системы

$$A_i T_{i+1}^{n+1} - B_i T_i^{n+1} + C_i T_{i-1}^{n+1} = D_i\,, \qquad i = \overline{2, N-1} \tag{4.5}$$

где $A_i = C_i = \frac{\lambda}{h^2},\ B_i = \frac{2\lambda}{h^2} + \frac{\rho C}{\tau},\ D_i = -\frac{\rho C}{\tau} T_i^n$

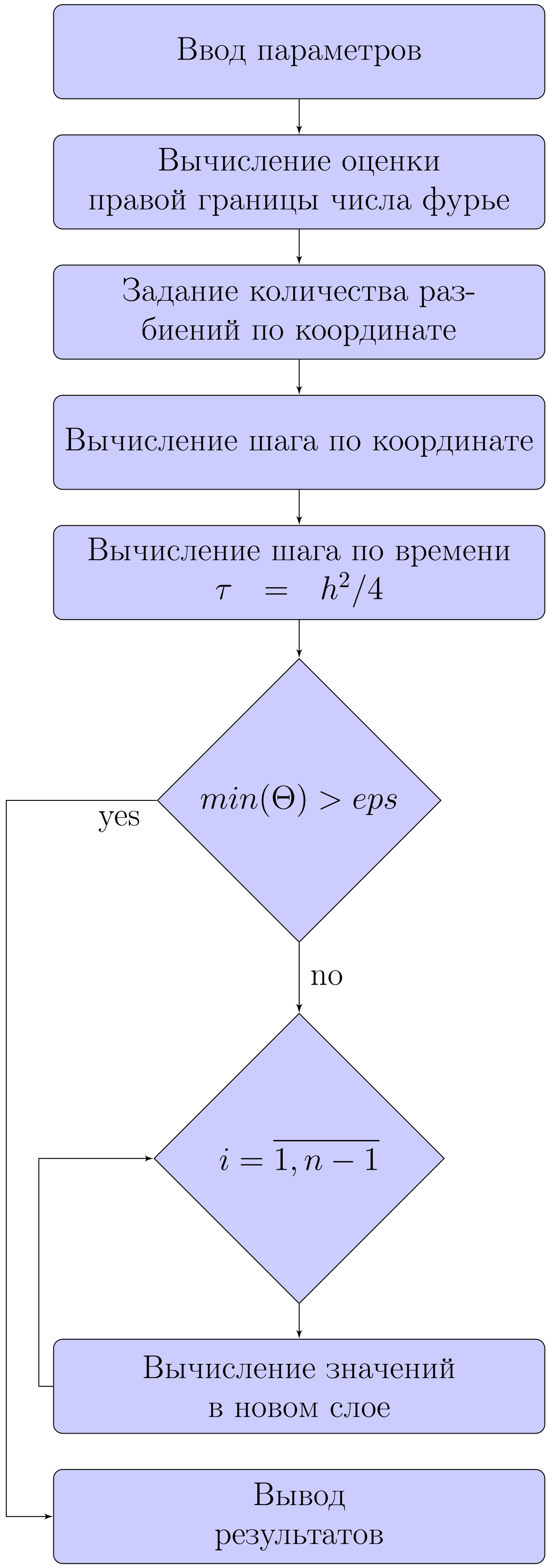


Рис. 4.2. Блок схема алгоритма конечных разностей (явный)

Эта система решается методом прогонки

$$\alpha_i = \frac{A_i}{B_i - C_i\alpha_{i-1}},\ \beta_i = \frac{C_i\beta_{i-1} - D_i}{B_i - C_i\alpha_{i-1}} \qquad i = \overline{2, N-1} \tag{4.6}$$

Аппроксимация дифференциальной задачи выполнена с первым порядком точности по времени t и вторым по пространственной координате h. При этом неявная разностная схема является абсолютно устойчивой, т.е. можн опроводить интегрирование краевой задачи с любым разностным шагом по времени. Шаг по времени выбирается таким образом,чтобы весь интервал времени разбивался хотя бы на 10 шагов, а при дроблении шага пополам полученные результаты отличались не более чем на $0,1 - 1\%$.

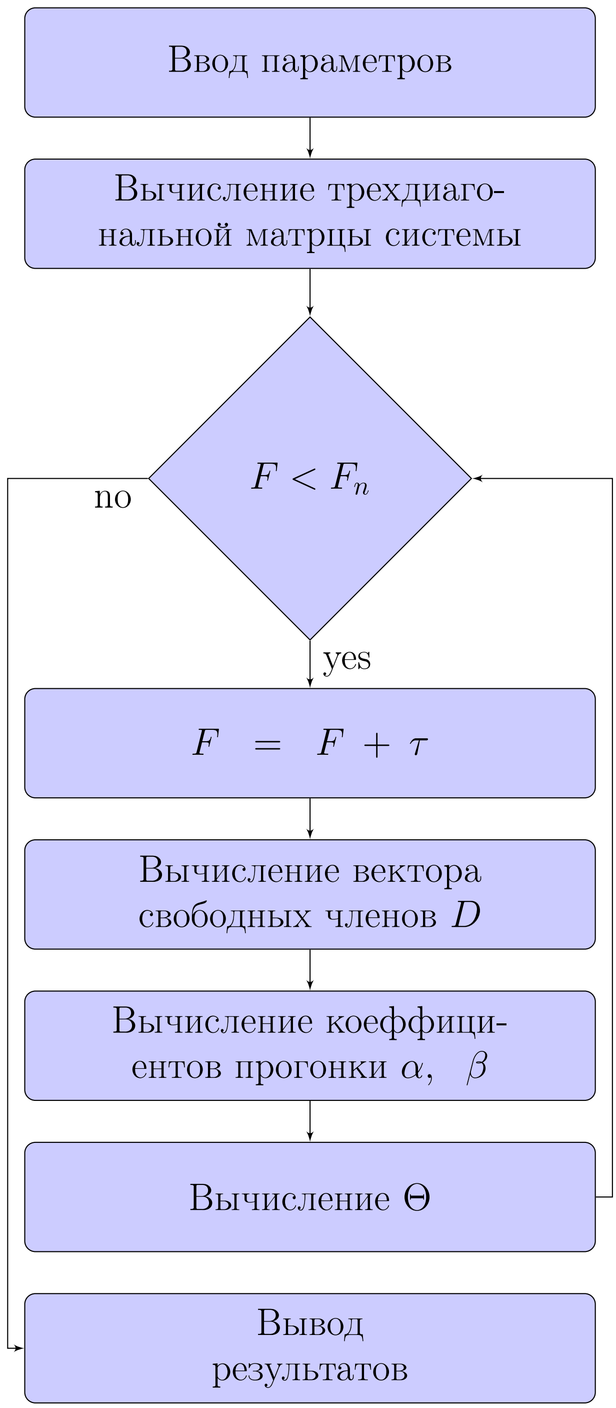


Рис. 4.3. Блок схема алгоритма конечных разностей (неявный)

V. Тестовый расчет

Номограммы $\Theta = f(Fo, Bi)$ для поверхности и середины тонкой пластины приведены на рис. 5.4 и 5.5 соответственно (Задачник по теплопередаче Краснощеков с 38-39).

(a) $Bi = 1$

Fo	Результат		Номограмма	
	$\xi = 0$	$\xi = 1$	$\xi = 0$	$\xi = 1$
1	0.5273	0.3456	0.5	0.4
2	0.2546	0.1660	0.25	0.19
3	0.1210	0.0789	0.12	0.09
4	0.0575	0.0375	0.06	0.04
5	0.0276	0.0180	0.027	0.2

(b) $Bi = 0.2$

Fo	Результат		Номограмма	
	$\xi = 0$	$\xi = 1$	$\xi = 0$	$\xi = 1$
3	0.5855	0.5315	0.58	0.54
5	0.4040	0.3668	0.4	0.35
8	0.2303	0.2091	0.22	0.21
12	0.1084	0.0984	0.105	0.1
20	0.02432	0.0220	0.025	0.25

Таблица 5.1. Метод разделения координат

(a) $Bi = 1$

Fo	Результат		Номограмма	
	$\xi = 0$	$\xi = 1$	$\xi = 0$	$\xi = 1$
1	0.5148	0.3703	0.5	0.4
2	0.2537	0.200	0.25	0.19
3	0.1346	0.1071	0.12	0.09
4	0.0671	0.0537	0.06	0.04
5	0.0278	0.0250	0.027	0.2

(b) $Bi = 0.2$

Fo	Результат		Номограмма	
	$\xi = 0$	$\xi = 1$	$\xi = 0$	$\xi = 1$
3	0.5779	0.5333	0.58	0.54
5	0.3823	0.3559	0.4	0.35
8	0.2308	0.21857	0.22	0.21
12	0.1106	0.1013	0.105	0.1
20	0.0257	0.0222	0.025	0.025

Таблица 5.2. Метод конечных разностей (явный)

(a) $Bi = 1$

Fo	Результат		Номограмма	
	$\xi = 0$	$\xi = 1$	$\xi = 0$	$\xi = 1$
1	0.5254	0.3430	0.5	0.4
2	0.2472	0.1614	0.25	0.19
3	0.1163	0.0759	0.12	0.09
4	0.0547	0.0498	0.06	0.04
5	0.0257	0.0168	0.027	0.02

(b) $Bi = 0.2$

Fo	Результат		Номограмма	
	$\xi = 0$	$\xi = 1$	$\xi = 0$	$\xi = 1$
3	0.5808	0.5273	0.58	0.54
5	0.3964	0.3598	0.4	0.35
8	0.2234	0.2028	0.22	0.21
12	0.1041	0.0944	0.105	0.1
20	0.0225	0.0204	0.025	0.025

Таблица 5.3. Метод конечных разностей (неявный)

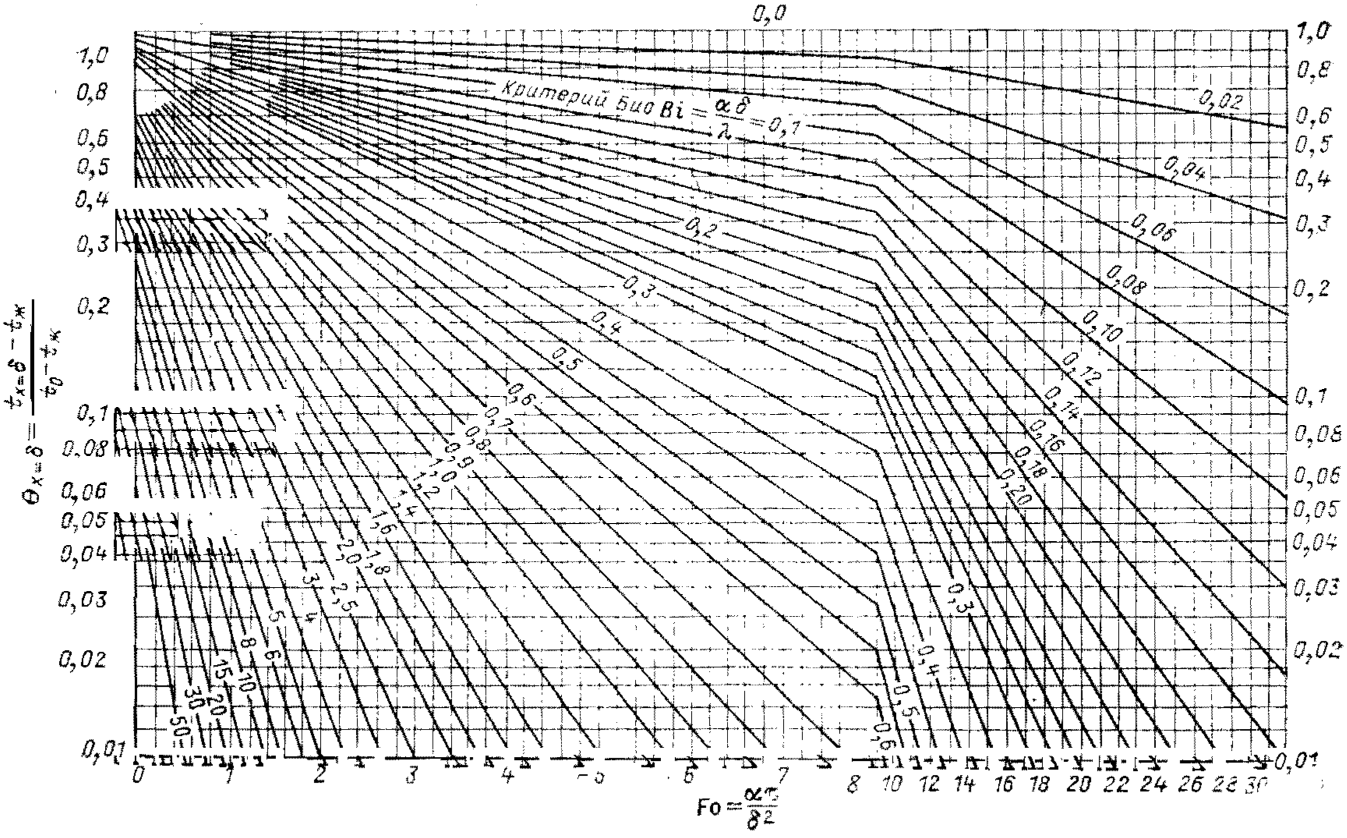


Рис. 5.4. Зависимость $\Theta = f(Fo, Bi)$ для поверхности тонкой пластины

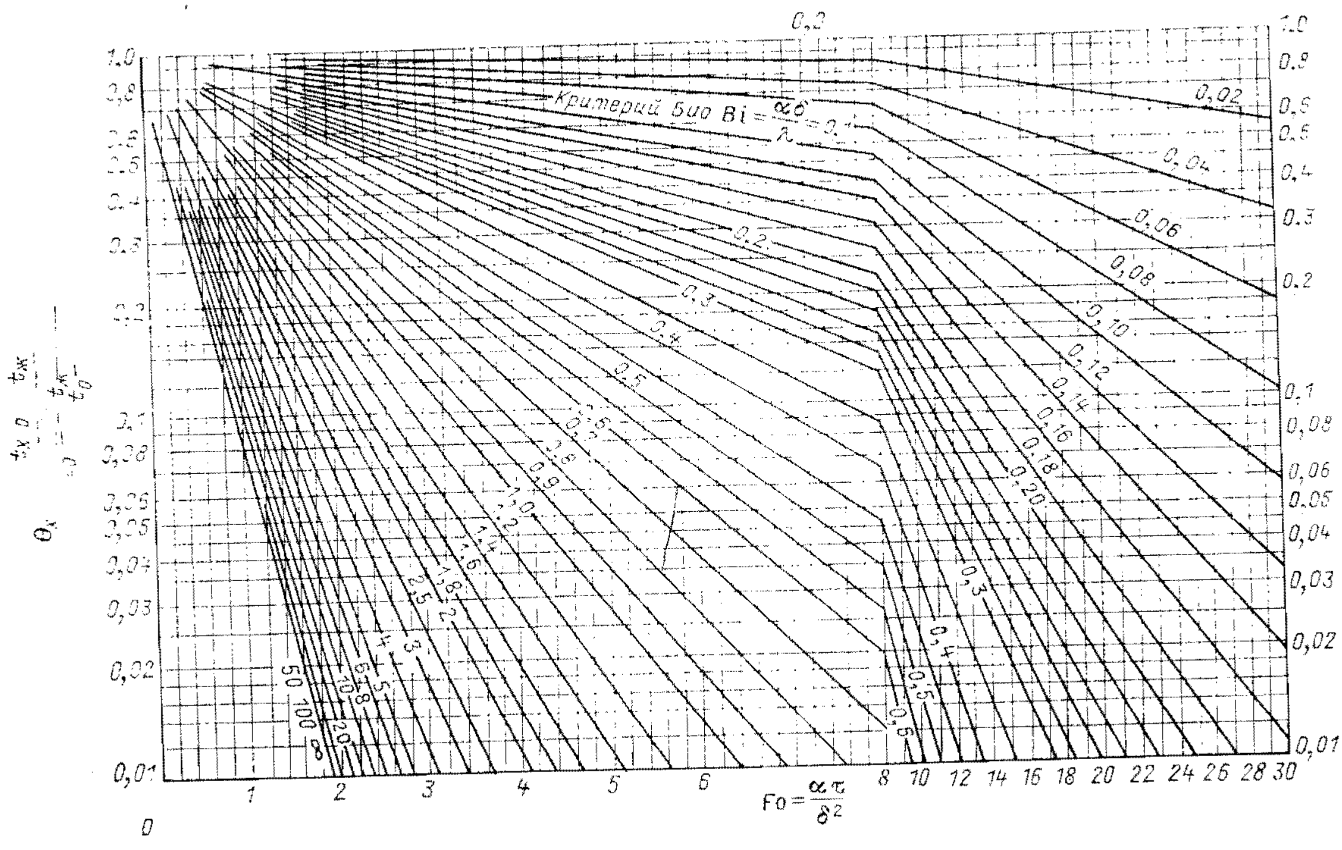
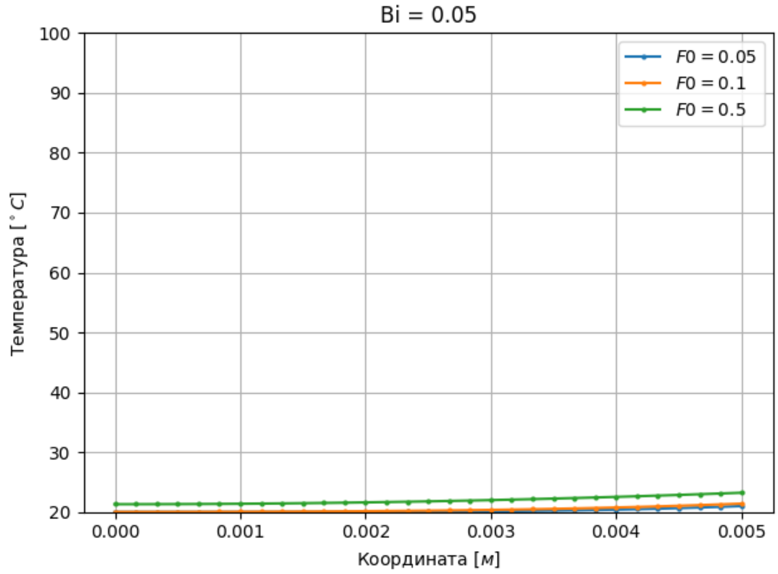


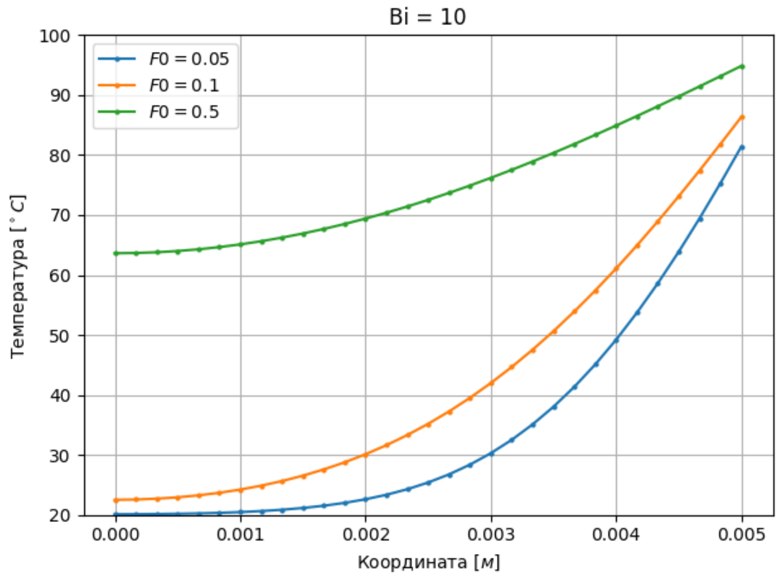
Рис. 5.5. Зависимость $\Theta = f(Fo, Bi)$ для середины тонкой пластины

VI. Результаты решения задачи

(a) $\alpha = 7.4$



(b) $\alpha = 1500$



(c) $\alpha = 11000$

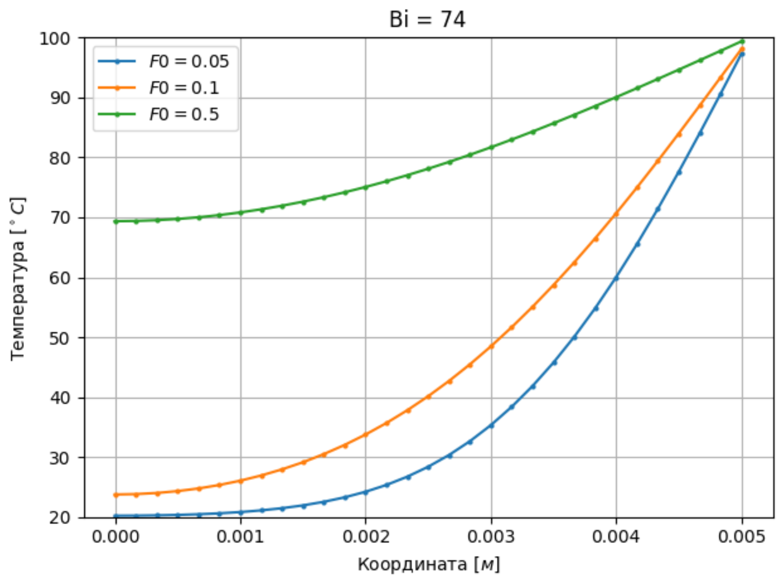
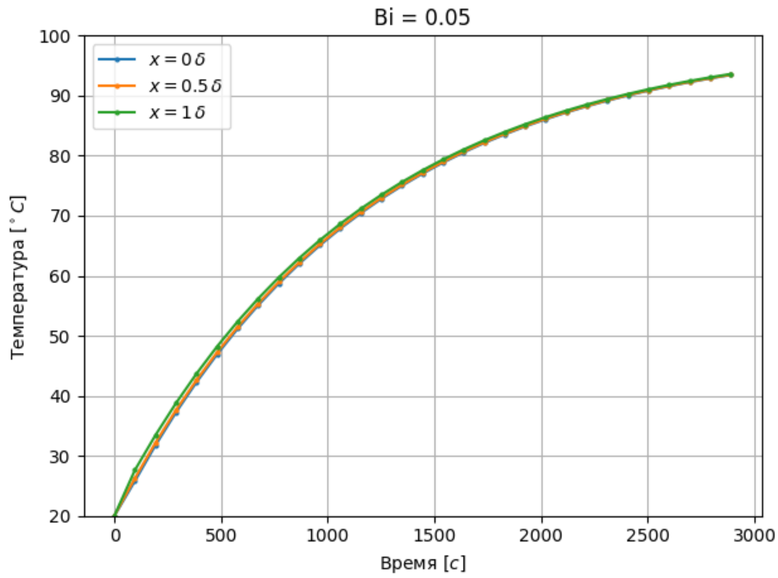


Рис. 6.1. Метод разделения координат

(a) $\alpha = 7.4$



(b) $\alpha = 11000$

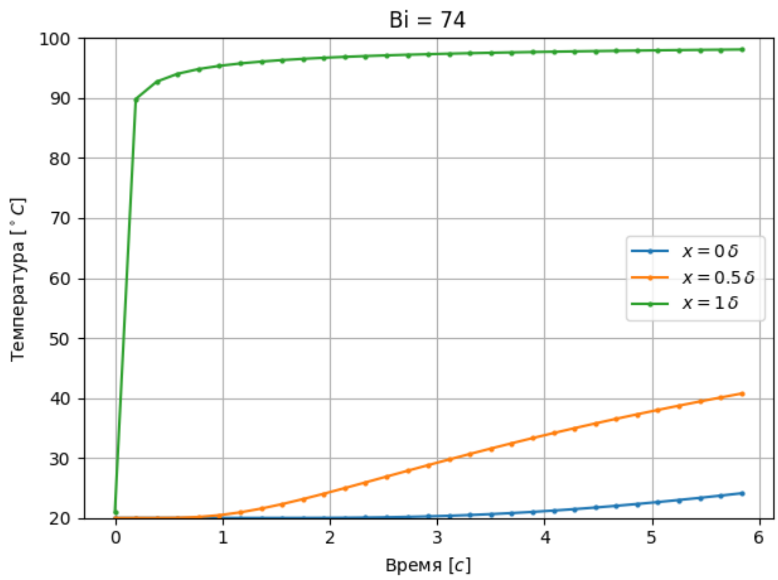
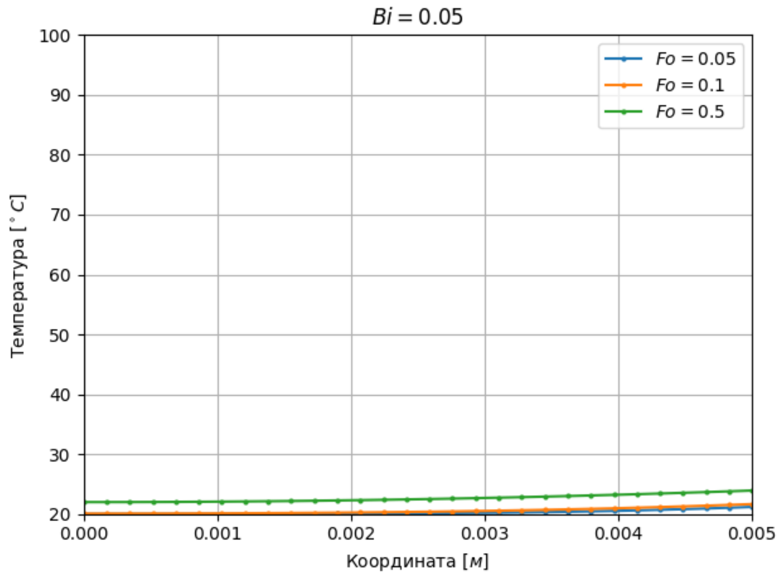
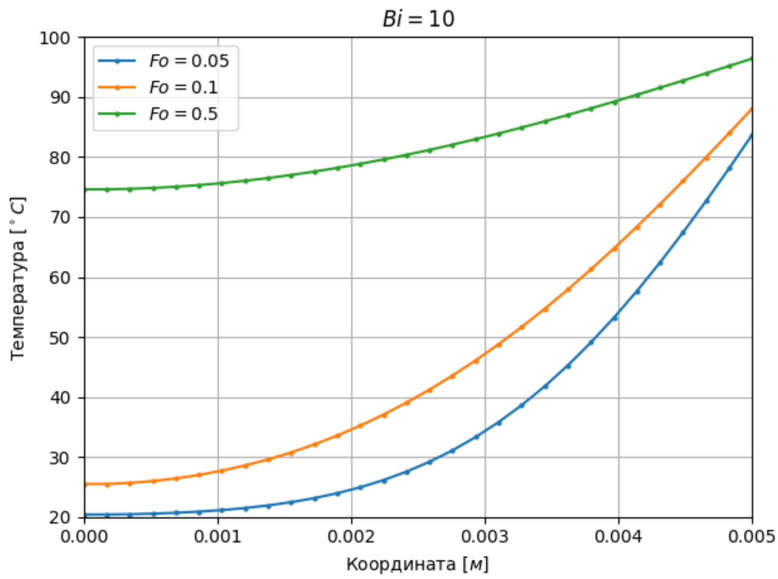


Рис. 6.2. Метод разделения координат

(a) $\alpha = 7.4$



(b) $\alpha = 1500$



(c) $\alpha = 11000$

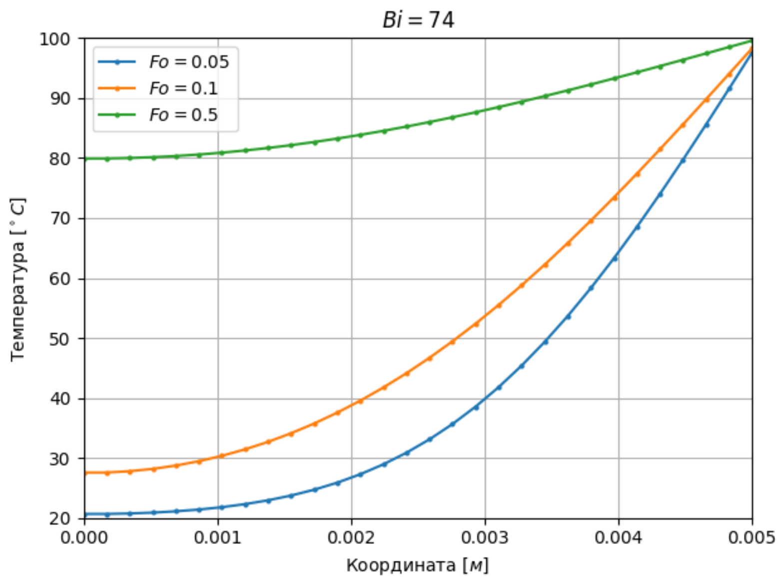
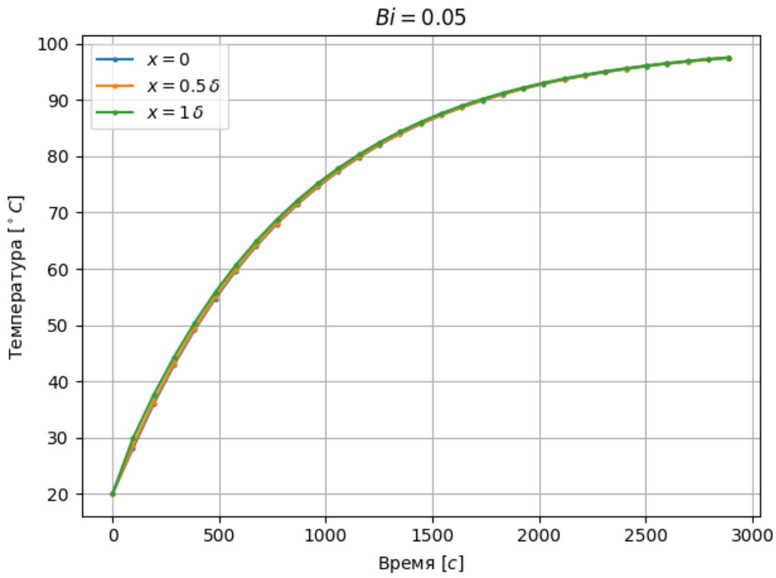


Рис. 6.3. Метод конечных разностей (явный)

(a) $\alpha = 7.4$



(b) $\alpha = 11000$

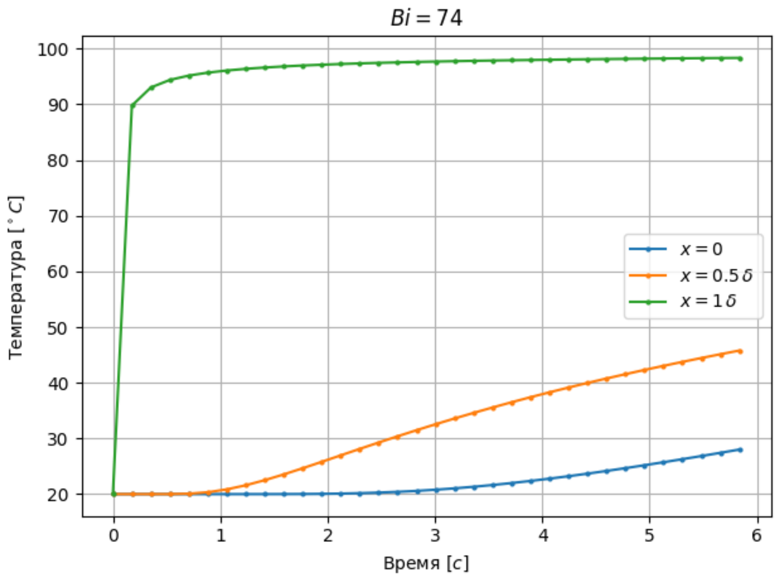
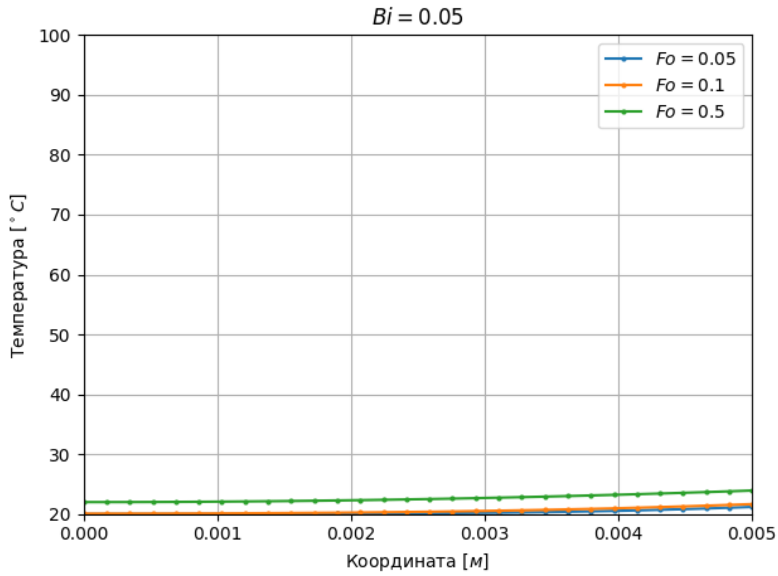
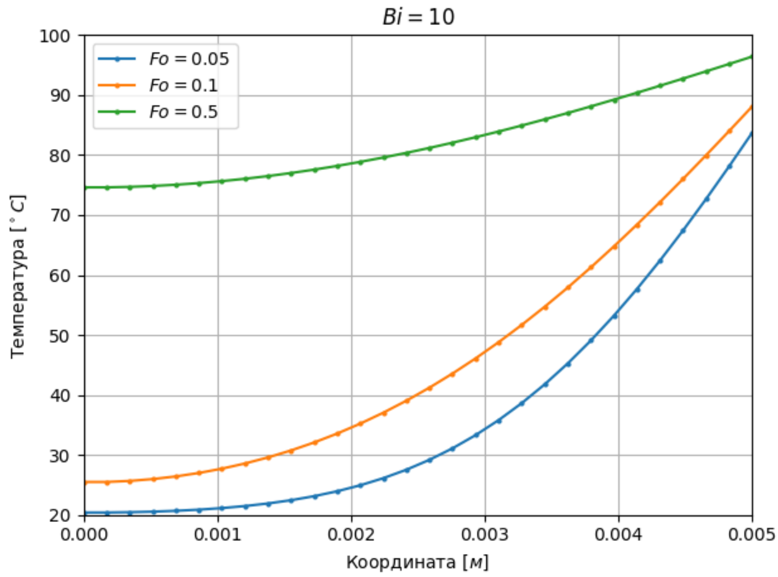


Рис. 6.4. Метод конечных разностей (явный)

(a) $\alpha = 7.4$



(b) $\alpha = 1500$



(c) $\alpha = 11000$

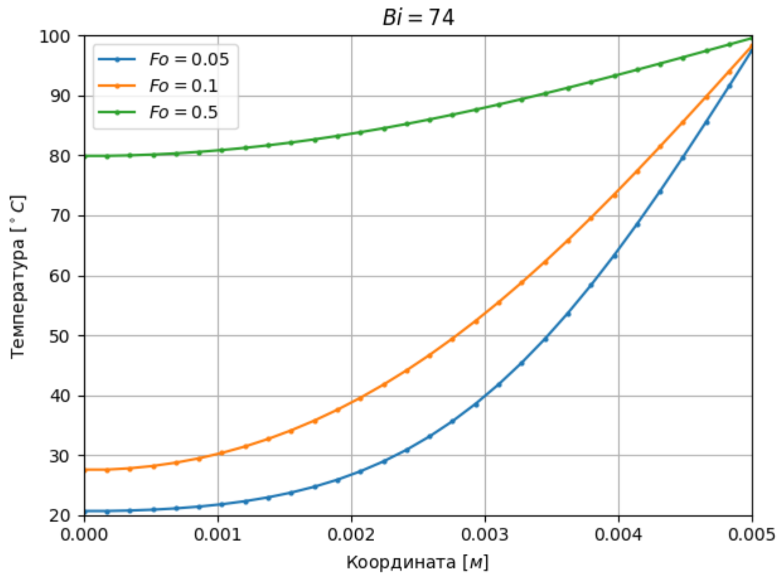
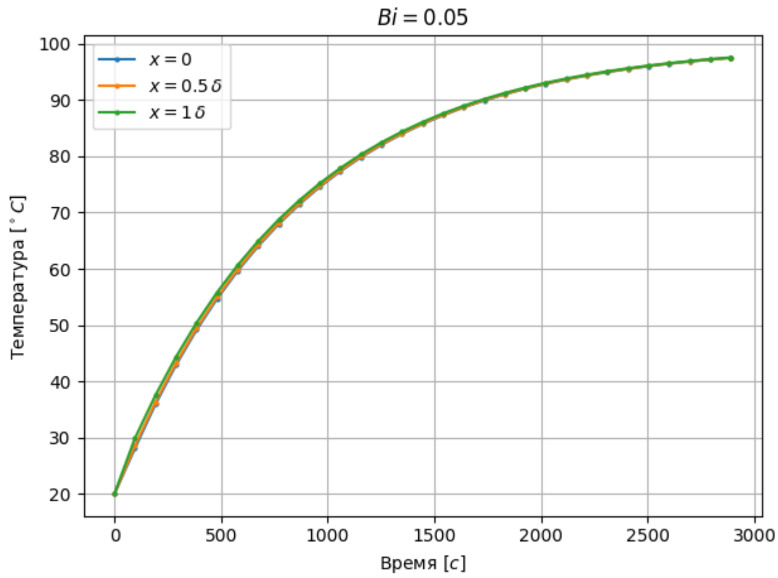


Рис. 6.5. Метод конечных разностей (неявный)

(a) $\alpha = 7.4$



(b) $\alpha = 11000$

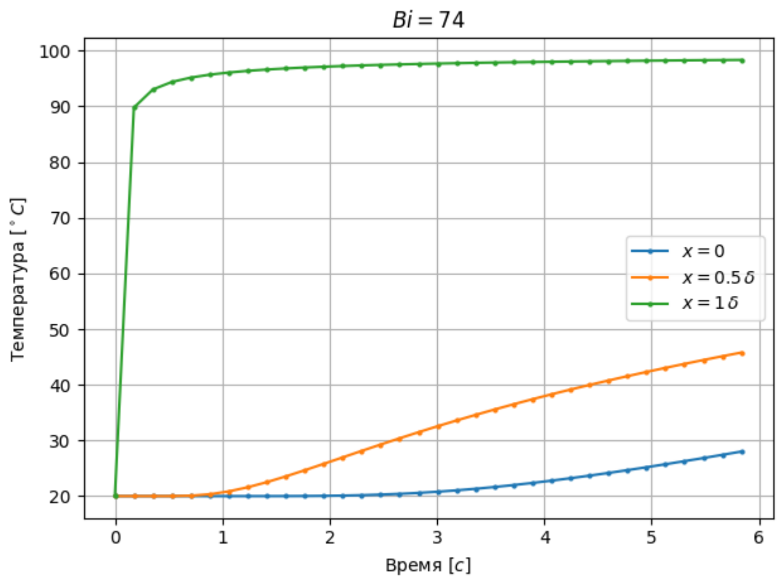


Рис. 6.6. Метод конечных разностей (неявный)

VII. Выводы

- **Отличие режимов теплообмена с малыми и большими значениями числа Био**
  Если толщина пластины и коэффициент теплопроводности фиксированы, то бóльшие числа Био будут соответствовать бóльшим значениям коэффициента конвективной теплоотдачи. С увеличением числа Био, время, за которое наступает тепловое равновесие, уменьшается.

- **Анализ количества членов ряда, необходимых для рассчета температуры с заданной точностью**
  При $Bi = 10$, $x = \delta$, $Fo = 0.05$ для достижения точности 5 знака потребовалось 6 членов ряда. При $Fo = 0.5$ потребовалось 3 члена. Следовательно, при уменьшении числа Фурье количество членов ряда, необходимое для достижения требуемой точности, растет.

- **Установление теплового равновесия тела с окружающей средой**
  Под установлением теплового равновесия будем понимать установление безразметной температуры в центре пластины меньше кого-либо малого ϵ. Возьмем $\epsilon = 10^{-1}$. При $Bi = 0.1$ равновесие наступает при числе Фурье 13. При $Bi = 100$ — $Fo = 2.7$. Т.е. с увеличением числа Био время наступления теплового равновесия уменьшается.

7.1. Выводы по результатам решения методом конечных разностей

- **Анализ влияния величины шагов по координате и времени на точность результатов**
  Сравнение решения, полученного методом конечных разностей с аналитическим представлены в таблице (7.1). Сравнивалась температура на поверхности пластины при $Fo = 0.05$, $Bi = 10$ с аналитическим решением $\Theta(Bi, Fo) = 20.1176$. Для достижения 3 значящих цифр достаточно 41 разбиения по координате.

Таблица 7.1. Сравнение решения, полученного методом конечных разностей с аналитическим, равным 20.1176

Количество разбиений по координате	Полученный результат для $x = \delta$
11	20.3578
21	20.2236
41	20.1438
81	20.1294

- **Анализ поведения алгоритма при невыполнении условия устойчивости**

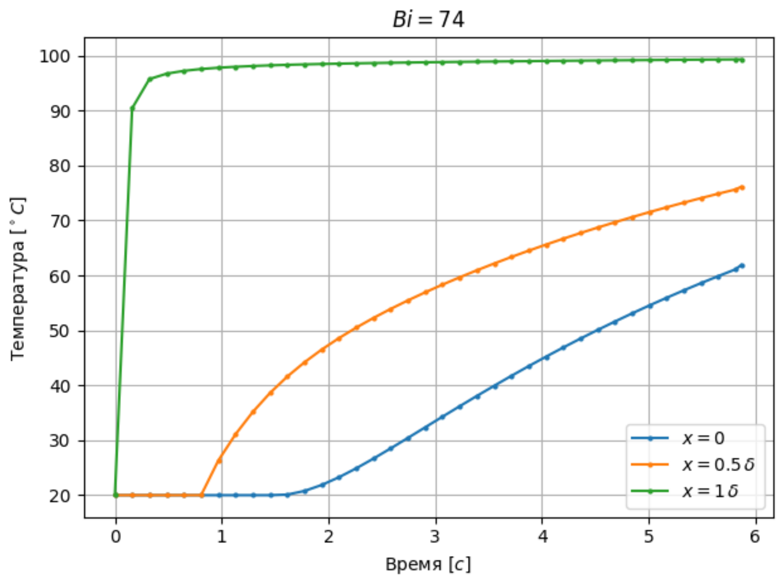


Рис. 7.1. Решение явным методом конечных разностей при невыполнении условия устойчивости

Проведем рассчет методом конечных разностей с явной схемой задав шаг по времени $\tau = 1.1\frac{h^2}{2}$.

Как видно, при невыполнении условия устойчивости метод конечных разностей с явной схемой работает некорректно.

- **Сравнеие трудоемкости программной реализации аналитического и конечно-разностного методов**
  Реализация аналитического метода значительно проще, так как задача сводиться к нахождению частичной суммы ряда с удовлетворяющей точностью. Но аналитическое решения доступно не для всех задач и не является универсальным способом решения краевых задач.

VIII. Приложение

8.1. Программа на языке Python

8.1.1. Метод разделения переменных

```
'''
Created on Mar 16, 2018

@author: godfather
'''
import math
import matplotlib.pylab as plt
import json


def f(x, Bi):
```

```
    return math.cos(x)/math.sin(x) - x/Bi


def bisection(a, b, eps, Bi):
    f1 = f(a, Bi)
    f2 = f(b, Bi)
    if f2*f1 > 0:
        print('Bad!') #предупреждение
    while True:
        c = (b+a)/2.0
        f3 = f(c, Bi)
        if f1*f3 < 0:
            b = c
        else:
            a = c
            f1 = f3
        if abs(a-b) < eps : break
    return c


def calc_T(eps, Bi, fourier_numb, dim_coord):
    '''
    Функция вычисляет температуру, как сумму ряда,
    при заданных координате и времени
    '''
    dim_T = 0
    n = 0
    while True:
        n += 1
        x1 = math.pi * (n - 1.0) + eps
        x2 = math.pi * (n - 0.5)
        mu = bisection(x1, x2, eps, Bi)
        an = 2.0 * math.sin(mu) / (mu + math.sin(mu) * math.cos(mu))
        dim_Tn = an * math.exp(-mu * mu * fourier_numb) * math.cos(mu * dim_coord
            )
        dim_T = dim_T + dim_Tn
        if abs(dim_Tn / dim_T) < eps: break #если следующий член ряда составляет
        # меньше eps от уже набранной суммы, то прекращаем цикл
    return dim_T


def ftime(fo):
    lam = 0.74
    c = 670
    ro = 2500

    a = lam / (c * ro)
    delta = 0.005

    return [delta**2 / a * i for i in fo]


def fteta(tet):
    return [ i * (20 - 100) + 100 for i in tet]


def temp_t(eps,Bi,dim_coord,a,b,numb):
    '''
    Функция строит зависимость температуры от времени при заданной координате
    '''
```

```
    h = (b - a) / numb
    x = []; y = []
    f = open('OutData/dim_coord='+str(dim_coord)+'.txt','w')
    for i in range(numb+1):
        fourier_numb = a + i*h;
        dim_T = calc_T(eps, Bi, fourier_numb, dim_coord)
        x.append(ftime([fourier_numb]))
        y.append(fteta([dim_T]))
        f.write(str(fourier_numb)+'\n\t'+str(dim_T)+'\n')
    f.close()
    plt.plot(x, y, '-', label='$x = {}\, \delta$'.format(dim_coord), marker='o', markersize=2)
    plt.legend()
    plt.title('Bi = '+str(Bi))
    plt.xlabel('Время [$c$]')
    plt.ylabel('Температура [$^\circ C$]')

def temp_coord(eps,Bi,a,b,fourier_numb, numb):
    '''
    Функция строит зависимость температуры от координаты при заданном моменте времени
    '''
    h = (b - a) / numb
    x = []; y = []
    f = open('OutData/fourier_numb='+str(fourier_numb)+'.txt', 'w')
    for i in range(numb+1):
        dim_coord = a + i*h;
        dim_T = calc_T(eps, Bi, fourier_numb, dim_coord)
        x.append(dim_coord*0.005)
        y.append(fteta([dim_T]))
        f.write(str(dim_coord)+'\n\t'+str(dim_T)+'\n')
    f.close()
    plt.plot(x, y, '-', label='$F0 = {}$'.format(fourier_numb), marker='o', markersize=2)
    plt.legend()
    plt.title('Bi = '+str(Bi))
    plt.xlabel('Координата [$м$]')
    plt.ylabel('Температура [$^\circ C$]')



def main():
    '''
    Функция представляет собой главный модуль, его вид
    следует менять в зависимости от поставленной задачи
    и входных данных.
    В данном слечае программа настроена на вывод 5 графиков,
    на кажном из которых представлено по 3 зависимости.
    Входные данные являются json файлом.
    '''
    for i in range(5):
        data = json.load(open('input_data.json'))[i]
        Bi = data['Bi']
        numb = data['numb']
        eps = data['eps']
        if i < 3 :
            dim_coord_0 = data['dim_cor_0']
            dim_coord_n = data['dim_cor_n']
            fig = plt.figure()
            for n in range(3):
```

```
                fourier_n = data['fourier_n'][n]
                temp_coord(eps, Bi, dim_coord_0, dim_coord_n, fourier_n, numb)
            fig.savefig('OutPlot/prog_1_temp_coord_Bi='+str(Bi)+'.png')
            plt.close(fig)
        else:
            fourier_0 = data['fourier_0']
            fourier_n = data['fourier_n']
            fig = plt.figure()
            for n in range(3):
                dim_coord_n = data['dim_cor_n'][n]
                temp_t(eps,Bi,dim_coord_n,fourier_n,fourier_0,numb)
            fig.savefig('OutPlot/prog_1_temp_t_Bi='+str(Bi)+'.png')
            plt.close(fig)

def mainn(Bi):
    fourier_0 = 0
    fourier_n = 4*Bi**(-0.85)
    eps = 10**(-5)
    dim_coord_n = [0, 0.5, 1]
    numb = 30

    fig = plt.figure()
    for x in dim_coord_n:
        temp_t(eps, Bi, x, fourier_n, fourier_0, numb)
    plt.ylim(20, 100)
    plt.grid()
    plt.show()
    fig.savefig('OutPlot/prog_1_temp_t_Bi=' + str(Bi).replace('.','') + '.png')
    plt.close(fig)

    fig = plt.figure()
    fourier_numb = [0.05, 0.1, 0.5]
    for x in fourier_numb:
        temp_coord(eps, Bi, 0, 1, x, numb)
    plt.ylim(20, 100)
    plt.grid()
    plt.show()
    fig.savefig('OutPlot/prog_1_temp_x_Bi=' + str(Bi).replace('.', '') + '.png')
    plt.close(fig)


if __name__ == '__main__':
    Bi = [0.05, 10, 74]
    for bi in Bi:
        mainn(bi)
```

8.1.2. Метод конечных разностей (явный)

```
'''
Created on Mar 30, 2018

@author: godfather
'''
import matplotlib.pylab as plt


def save_to_plot(teta,i):
    global x1, x1n, xn, f
```

```
    x1.append(teta[0])
    x1n.append(teta[int(nx / 2)])
    xn.append(teta[int(nx - 1)])
    f.append(i * hf)


def export_to_txt(x1, x1n, xn, f, bio):
    file = open('konechno_razn_bio='+str(bio)+'.txt','w')
    form = '{:18.15f} '
    form_head = '{:^18} '
    out_str =''
    head = ''
    for i in range(4):
        out_str += form
        head += form_head
    out_str += '\n'
    head += '\n'
    file.write(head.format('fourie numb','x = 0','x = 0.5','x = 1'))
    for i in range(4*18+3):
        file.write('_')
    file.write('\n')
    for i in range(len(f)):
        file.write(out_str.format(f[i], x1[i], x1n[i], xn[i]))
    for i in range(4*18+3):
        file.write('_')
    file.write('\n')
    file.close()

def razm(teta1, teta12, teta2, time, bio, fb, xteta, dx, n):
    lam = 0.74
    c = 670
    ro = 2500

    a = lam /(c * ro)
    delta = 0.005

    def ftime(fo):
            return [delta**2 / a * i for i in fo]

    def fteta(tet):
        return [ i * (20 - 100) + 100 for i in tet]

    fb = ftime([fb])[0]
    time = ftime(time)
    teta1 = fteta(teta1)
    teta12 = fteta(teta12)
    teta2 = fteta(teta2)


    # my_plot(teta1, teta12, teta2, time, bio, fb)

    x = [i*dx*delta for i in range(n)]
    temp = [fteta(xtet) for xtet in xteta]
    my_plot2(temp, x, bio, delta)

def my_plot(x1, x1n, xn, f, bio, fb):
    fig = plt.figure()
    plt.plot(f, x1, '-', label='$x = 0$', marker='o', markersize=2)
    plt.plot(f, x1n, '-', label='$x = 0.5\, \delta$', marker='o', markersize=2)
    plt.plot(f, xn, '-', label='$x = 1\, \delta$', marker='o', markersize=2)
```

```
    plt.grid()
    plt.legend()
    plt.title('$Bi = {}$'.format(bio))
    plt.xlabel('Время [$c$]')
    plt.ylabel('Температура [$^\circ C$]')
    plt.axes([0, fb, 20, 100])
    plt.show()
    fig.savefig('OutPlot/prog_2_temp_t_Bi=' + str(bio).replace('.','') + '.png')
    plt.close(fig)


def my_plot2(xtet, x, bio, delta):
    fig = plt.figure()
    plt.plot(x, xtet[0], '-', label='$Fo = 0.05$', marker='o', markersize=2)
    plt.plot(x, xtet[1], '-', label='$Fo = 0.1$', marker='o', markersize=2)
    plt.plot(x, xtet[2], '-', label='$Fo = 0.5$', marker='o', markersize=2)
    plt.legend()
    plt.title('$Bi = {}$'.format(bio))
    plt.xlabel('Координата [$м$]')
    plt.ylabel('Температура [$^\circ C$]')
    plt.xlim(0, delta)
    plt.ylim(20, 100)
    plt.grid()
    plt.show()
    fig.savefig('OutPlot/prog_2_temp_x_Bi=' + str(bio).replace('.','') + '.png')
    plt.close(fig)


def main(bio):
    global x1, x1n, xn, f, nx, hf
    x1 = [1]
    x1n = [1]
    xn = [1]
    f = [0]
    bfur = 4 * bio ** (-0.85)
    nx = 30
    nf_plot = 30

    hx = 1.0 / (nx - 1.0)
    hf = hx ** 2 / 4

    nf_save = int(bfur / (hf * nf_plot))

    teta = []
    for i in range(nx):
        teta.append(1.0)

    # for i in range(int(nf)):
    i = 0
    # while min(teta)>eps:
    sechenie = []
    while i * hf < 0.6: #bfur:
        i += 1
        tet = teta

        for t in [0.05, 0.1, 0.5]:
            if abs(i * hf - t) < hf / 2:
                sechenie.append([el for el in teta])

        if i % nf_save == 0:
```

```
            save_to_plot(teta, i)

        for k in range(1, int(nx - 1)):
            teta[k] = ((1.0 - 2.0 * hf / (hx * hx)) * tet[k] + (hf / (hx * hx)) *
                (tet[k + 1] + tet[k - 1]))

        teta[0] = teta[1]
        teta[int(nx - 1)] = teta[int(nx - 2)] / (1.0 + hx * bio)

    if i % nf_save != 0:
        save_to_plot(teta, i)

    export_to_txt(x1, x1n, xn, f, bio)
    # my_plot(x1, x1n, xn, f, bio)
    razm(x1, x1n, xn, f, bio, bfur, sechenie, hx, nx)


if __name__ == '__main__':
    bio = [0.05, 10, 74]
    for i in range(3):
        main(bio[i])
    print('Complete!')
```

8.1.3. Метод конечных разностей (неявный)

```
import matplotlib.pyplot as plt


def export_to_txt(x1, x1n, xn, f, bio):
    file = open('konechno_razn_bio='+str(bio)+'.txt','w')
    form = '{:18.15f} '
    form_head = '{:^18} '
    out_str =''
    head = ''
    for i in range(4):
        out_str += form
        head += form_head
    out_str += '\n'
    head += '\n'
    file.write(head.format('fourie numb','x = 0','x = 0.5','x = 1'))
    for i in range(4*18+3):
        file.write('_')
    file.write('\n')
    for i in range(len(f)):
        file.write(out_str.format(f[i], x1[i], x1n[i], xn[i]))
    for i in range(4*18+3):
        file.write('_')
    file.write('\n')
    file.close()


def my_plot(x1, x1n, xn, f, bio, fb):
    fig = plt.figure()

    plt.plot(f, x1, '-', label='$x = 0$', marker = 'o', markersize=2)
    plt.plot(f, x1n, '-', label='$x = 0.5\, \delta$', marker = 'o', markersize=2)
    plt.plot(f, xn, '-', label='$x = 1\, \delta$', marker = 'o', markersize=2)
    plt.legend()
```

```
    plt.grid()
    plt.title('$Bi = {}$'.format(bio))
    plt.xlabel('Время [$c$]')
    plt.ylabel('Температура [$^\circ C$]')
    plt.axes([0, fb, 20, 100])
    plt.show()
    fig.savefig('OutPlot/prog_3_temp_t_Bi=' + str(bio).replace('.','') + '.png')
    plt.close(fig)


def my_plot2(xtet, x, bio, delta):
    fig = plt.figure()
    print(x)
    print(xtet[0])
    print(xtet[1])
    plt.plot(x, xtet[0], '-', label='$Fo = 0.2$', marker='o', markersize=2)
    plt.plot(x, xtet[1], '-', label='$Fo = 1$', marker='o', markersize=2)
    plt.plot(x, xtet[2], '-', label='$Fo = 2$', marker='o', markersize=2)
    plt.legend()
    plt.title('$Bi = {}$'.format(bio))
    plt.xlabel('Время [$c$]')
    plt.ylabel('Температура [$^\circ C$]')
    plt.xlim(0, delta)
    plt.ylim(20, 100)
    plt.grid()
    plt.show()
    fig.savefig('OutPlot/prog_3_temp_x_Bi=' + str(bio) + '.png')
    plt.close(fig)

def razm(teta1, teta12, teta2, time, bio, fb, xteta, dx, n):
    lam = 0.74
    c = 670
    ro = 2500

    a = lam /(c * ro)
    delta = 0.005

    def ftime(fo):
            return [delta**2 / a * i for i in fo]

    def fteta(tet):
        return [ i * (20 - 100) + 100 for i in tet]

    fb = ftime([fb])[0]
    time = ftime(time)
    teta1 = fteta(teta1)
    teta12 = fteta(teta12)
    teta2 = fteta(teta2)


    # my_plot(teta1, teta12, teta2, time, bio, fb)

    x = [i*dx*delta for i in range(n)]
    temp = [fteta(xtet) for xtet in xteta]
    # print(len(temp[2]))
    # print(len(x))
    my_plot2(temp, x, bio, delta)

def main(bio):
    n = 30
```

```
    furie_0 = 0

    furie_n = 4*bio**(-0.85)
    dx = 1/(n-1)
    h = dx**2/1.5

    teta = [1 for i in range(n)]
    teta1, teta12, teta2 = [1], [1], [1]
    time = [0]

    Ak = [1/dx]
    Ak.extend([1/dx**2 for i in range(2, n)])
    Ak.append(0)
    Bk = [-1/dx]
    Bk.extend([-2/dx**2 - 1/h for i in range(2, n)])
    Bk.append(-bio - 1/dx)
    Ck = [0]
    Ck.extend([1/dx**2 for i in range(2, n)])
    Ck.append(1/dx)
    Dk = [0 for i in range(n)]
    tetax = []
    furie = furie_0

    k = 0
    while furie < 1: #furie_n:
        k += 1
        furie += h
        Dk[1:n-2] = map(lambda x: x/h, teta[1:n-2])
        alpha = [-Ak[0]/Bk[0]]
        beta = [-Dk[0]/Bk[0]]

        for i in range(1, n-1):
            alpha.append(-Ak[i]/(Bk[i] + alpha[i-1]*Ck[i]))
            beta.append(-(Dk[i] + Ck[i]*beta[i-1])/(Bk[i] + alpha[i-1]*Ck[i]))

        beta.append(-(Ck[n-1]*beta[n-2])/(Bk[n-1]+alpha[n-2]*Ck[n-1]))
        teta[n-1] = beta[n-1]

        for i in range(n-2, -1, -1):
            teta[i] = alpha[i]*teta[i+1] + beta[i]

        for i in [0.05, 0.1, 0.5]:
            if abs(furie-i)<h/2:
                tetax.append([el for el in teta])

        if k % int(furie_n/(30*h)) == 0:
            teta1.append(teta[0])
            teta12.append(teta[int((n-1)/2)])
            teta2.append(teta[n-1])
            time.append(furie)

    export_to_txt(teta1, teta12, teta2, time, bio)
    razm(teta1, teta12, teta2, time, bio, furie_n, tetax, dx, n)


if __name__=='__main__':
    bio = [0.05, 10, 74]
    for i in bio:
        main(i)
    print('Complete!')
```