Санкт-Петербургский государственный университет
Математическое обеспечение и администрирование информационных систем
Параллельное программирование

Смирнов Илья Евгеньевич

Применение сплайнов для решения уравнения теплопроводности и распараллеливание

Выпускная квалификационная работа

Научный руководитель:
д.ф.-м.н., профессор Бурова И.Г.

Рецензент:
старший преподаватель Мирошниченко И.Д.

Санкт-Петербург
2017

SAINT-PETERSBURG STATE UNIVERSITY
Software and Administration of Information Systems
Parallel Programming

Ilia Smirnov

Application of splines for solving heat conduction equation and paralleling

Graduation Project

Scientific supervisor:
Irina Burova

____

Reviewer:
Irina Miroshnichenko

____

Saint-Petersburg
2017

Содержание

1 Введение

В данной работе рассматривается уравнение теплопроводности с двумя независимыми переменными

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \qquad (1),$$

где $u = u(x,t)$ — искомая функция переменных x и t. Это уравнение — линейное дифференциальное уравнение второго порядка параболического типа. Далее рассматривается задача отыскания решений уравнения, определенных в замкнутом прямоугольнике плоскости

$$Q = \{(x,t) : a \leq x \leq b, 0 \leq t \leq T\}$$

Дифференциальное уравнение имеет бесконечно много решений. Для того, чтобы из этого множества выделить одно решение, надо задать дополнительную информацию об искомом решении. Обычно такая информация задается в виде начального условия

$$u(x,0) = u_0(x) \qquad (2)$$

и краевых условий

$$u(a,t) = \psi_1(t), u(b,t) = \psi_2(t). \qquad (3)$$

Уравнение теплопроводности является наиболее простым из уравнений параболического типа (см. [1], [2], [3]). Несмотря на простоту уравнения (1), его решение также является достаточно трудоёмким, особенно в случаях, когда необходима большая точность. К тому же, опыт, полученный при работе с данным уравнением, вероятно, можно перенести и на другие параболические уравнения. Обычно в вычислительных методах для этих целей применяется метод сеток [4], [5] и разностные формулы для аппроксимации производных. В этой работе будут рассмотрены несколько разных подходов к этому методу, в том числе с использованием сплайнов [6], а также то, как распараллеливание влияет на скорость вычислений [7]. Наша задача — сократить время вычислений, применяя аппарат распараллеливания OpenMP.

2 Цель работы

Разработать программу на языке C++ для численного решения задачи.

Предложить два варианта распараллеливания вычислений.

Провести численные эксперименты, визуализировать результаты.

3 Аппроксимация сплайнами

Рассматривается задача теплопроводности

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \tag{1}$$

с начальным условием

$$u(x, 0) = u_0(x)$$

и краевыми условиями

$$u(a, t) = \phi_1(t), u(b, t) = \phi_2(t)$$

в области

$$Q = \{(x, t) : a \le x \le b, 0 \le t \le T\},$$

где $u = u(x, t)$ — искомая функция переменных x и t.

Построим сетку узлов в полосе Q с шагом h по x и шагом τ по t, $h > 0$, $\tau > 0$.

Введем обозначения: $x_j = a + hj$, $t_k = k\tau$, u_j^k - значение u в точке (x_j, t_k).

Построим разностную схему с использованием элементов теории аппроксимации сплайнами для второй производной по x.

Для данного алгоритма была использована формула

$$u^{(\alpha)}(x, t_k) \simeq \omega_{j-1}^{(\alpha)}(x)u(x_{j-1}, t_k) + \omega_j^{(\alpha)}(x)u(x_j, t_k) + \omega_{j+1}^{(\alpha)}(x)u(x_{j+1}, t_k), \tag{2}$$

где $x \in [x_j, x_{j+1}]$, $\omega_j(x), \omega_{j-1}(x), \omega_{j+1}(x)$ — базисные сплайны, α — 1, 2.

Из формулы (2)

$$\left.\frac{\partial^2 u}{\partial t^2}\right|_{(x_j,t_k)} \simeq \omega_{j-1}^{''} u_{j-1}^{k-1} + \omega_j^{''} u_j^{k-1} + \omega_{j+1}^{''} u_{j+1}^{k-1}. \tag{3}$$

Разностная формула для первой производной

$$\left.\frac{\partial u}{\partial t}\right|_{(x_j,t_k)} \simeq \frac{u_j^k - u_j^{k-1}}{\tau}. \tag{4}$$

Таким образом, из формул (3) и (4) получаем следующую схему:

$$u_j^k = \tau(\omega_{j-1}^{''} u_{j-1}^{k-1} + \omega_{j+1}^{''} u_{j+1}^{k-1}) + (1 + \tau\omega_j^{''})u_j^{k-1}. \tag{5}$$

Рассмотрим квадратичный полиномиальный сплайн (см. [2]).

Базисные сплайны вычисляются по формуле

$$\omega_j(x) = \begin{cases} \dfrac{(x - x_{j+1})(x - x_{j+2})}{(x_j - x_{j+1})(x_j - x_{j+2})}, x \in [x_{j+1}, x_{j+2}], \\ \dfrac{(x - x_{j-1})(x - x_{j+1})}{(x_j - x_{j-1})(x_j - x_{j+1})}, x \in [x_j, x_{j+1}], \\ \dfrac{(x - x_{j-2})(x - x_{j-1})}{(x_j - x_{j-2})(x_j - x_{j-1})}, x \in [x_{j-1}, x_j]. \end{cases} \tag{6}$$

Вычислим вторую производную из формулы (6):

$$\omega_j^{''}(x) = \begin{cases} \dfrac{2}{(x_j - x_{j+1})(x_j - x_{j+2})}, x \in [x_{j+1}, x_{j+2}], \\ \dfrac{2}{(x_j - x_{j-1})(x_j - x_{j+1})}, x \in [x_j, x_{j+1}], \\ \dfrac{2}{(x_j - x_{j-2})(x_j - x_{j-1})}, x \in [x_{j-1}, x_j]. \end{cases} \tag{7}$$

Получаем, что при $x \in [x_j, x_{j+1}]$

$$\omega_{j-1}^{''}(x) = \frac{1}{h^2},\ \omega_j^{''}(x) = -\frac{2}{h^2},\ \omega_{j+1}^{''}(x) = \frac{1}{h^2}. \tag{8}$$

Для сравнения будем использовать также тригонометрический сплайн.

Базисные сплайны выглядят следующим образом:

$$\omega_j(x) = \begin{cases} \dfrac{sin(\frac{x-x_{j-1}}{2})sin(\frac{x-x_{j-2}}{2})}{sin(\frac{x_j-x_{j-1}}{2})sin(\frac{x_j-x_{j-2}}{2})}, x \in [x_{j-1}, x_j], \\ \dfrac{sin(\frac{x-x_{j-1}}{2})sin(\frac{x-x_{j+1}}{2})}{sin(\frac{x_j-x_{j-1}}{2})sin(\frac{x_j-x_{j+1}}{2})}, x \in [x_j, x_{j+1}], \\ \dfrac{sin(\frac{x-x_{j+1}}{2})sin(\frac{x-x_{j+2}}{2})}{sin(\frac{x_j-x_{j+1}}{2})sin(\frac{x_j-x_{j+2}}{2})}, x \in [x_{j+1}, x_{j+2}]. \end{cases} \tag{9}$$

Посчитаем вторую производную из формулы (9).

$$\omega_j^{''}(x) = \begin{cases} \dfrac{cos(x - \frac{x_{j-1}+x_{j-2}}{2})}{cos(\frac{x_{j-1}-x_{j-2}}{2}) - cos(x_j - \frac{x_{j-1}+x_{j-2}}{2})}, x \in [x_{j-1}, x_j], \\ \dfrac{cos(x - \frac{x_{j-1}+x_{j+1}}{2})}{cos(\frac{x_{j-1}-x_{j+1}}{2}) - cos(x_j - \frac{x_{j-1}+x_{j+1}}{2})}, x \in [x_j, x_{j+1}], \\ \dfrac{cos(x - \frac{x_{j+1}+x_{j+2}}{2})}{cos(\frac{x_{j+1}-x_{j+2}}{2}) - cos(x_j - \frac{x_{j+1}+x_{j+2}}{2})}, x \in [x_{j+1}, x_{j+2}]. \end{cases} \tag{10}$$

Таким образом на отрезке $[x_j, x_{j+1}]$ имеем

$$\begin{aligned} \omega''_{j-1}(x) &= \frac{1}{2(1-cos(h))}, \\ \omega''_{j}(x) &= -\frac{1}{1-cos(h)}, \\ \omega''_{j+1}(x) &= \frac{1}{2(1-cos(h))}. \end{aligned} \tag{11}$$

Нетрудно видеть, что при $h \to 0$

$$\omega''_{j-1}(x) = \frac{1}{h^2},\ \omega''_{j}(x) = -\frac{2}{h^2},\ \omega''_{j+1}(x) = \frac{1}{h^2}.$$

4 Явная схема метода сеток

Рассмотрим явную схему для решения уравнения теплопроводности. Один из главных недостатков вытекает из условия устойчивости

$$\tau \le \frac{h^2}{2}. \tag{12}$$

Из условия устойчивости вытекает, что возможно задавать только сравнительно небольшой шаг по t, что помогает составить более полную физическую картину, но замедляет работу, что делает распараллеливание особенно важным.

Чаще всего в методе сеток для аппроксимации производных используют разностные формулы:

$$\left.\frac{\partial u}{\partial t}\right|_{(x_j,t_k)} \simeq \frac{u_j^k - u_j^{k-1}}{\tau}, \tag{13}$$

$$\left.\frac{\partial^2 u}{\partial t^2}\right|_{(x_j,t_k)} \simeq \frac{u_{j+1}^{k-1} - 2u_j^{k-1} + u_{j-1}^{k-1}}{h^2}. \tag{14}$$

Таким образом, из формул (13) и (14) получается явная схема метода сеток

$$u_j^k = \frac{\tau}{h^2}\left(u_{j-1}^{k-1} + u_{j+1}^{k-1}\right) + \left(1 - 2\frac{\tau}{h^2}\right) u_j^{k-1}. \tag{15}$$

Используя результаты предыдущего параграфа, также можно получить две разностные схемы.

Явно запишем схему с полиномиальным сплайном, подставив (8) в (5):

$$u_j^k = \frac{\tau}{h^2}\left(u_{j-1}^{k-1} + u_{j+1}^{k-1}\right) + \left(1 - 2\frac{\tau}{h^2}\right) u_j^{k-1}. \tag{16}$$

Видно, что схема (16) совпадает со схемой (15).

Аналогично явную схему можно представить в следующем виде, подставив (11) в (5):

$$u_j^k = \left(\frac{\tau}{2(1 - cos(h))}\right)\left(u_{j-1}^{k-1} + u_{j+1}^{k-1}\right) + \left(1 - \frac{\tau}{1 - cos(h)}\right) u_j^{k-1} \tag{17}$$

Реализация явной схемы на языке C++ представлена в Listing 1.

```
Listing 1:
for (int k = 0; k < m - 1; k++)
        for (int i = 1; i < n; i++)
                u[i][k + 1] = koef1 * (u[i + 1][k] + u[i
                  - 1][k]) + koef2 * u[i][k];
```

В Listing 1 u — массив искомых значений, в котором заданы начальные и граничные условия,n и m — число узлов сетки по x и t соответственно. Если используется схема (16), koef1 и koef2 вычисляются как представлено в Listing 2, а если используется схема (17), то как представлено в Listing 3.

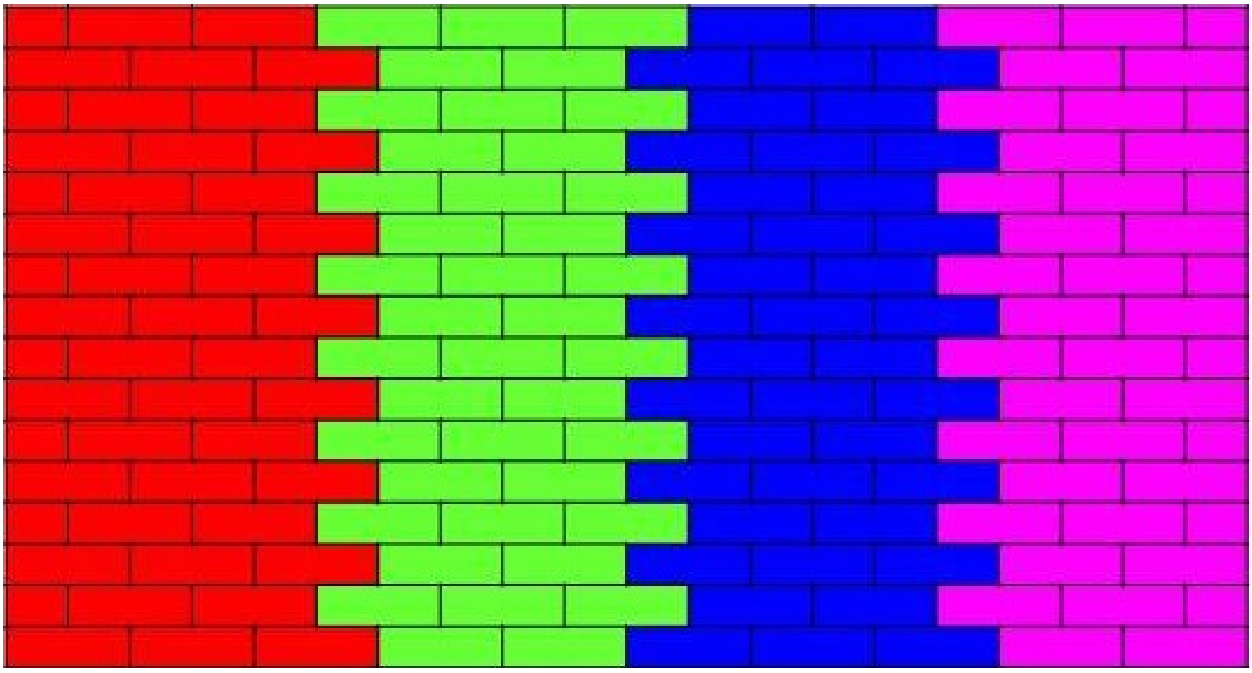
```
Listing 2:
tmp = tau / h / h;
koef1 = tmp;
koef2 = 1 - 2 * tmp;
```

```
Listing 3:
tmp = tau / (1 - cos(h));
koef1 = tmp / 2;
koef2 = 1 - tmp;
```

В Listing 2 и Listing 3 tau и h - шаги сетки соответственно по t и x.

5 Распараллеливание

Распараллеливания алгоритма выполнялось с использованием библиотеки OpenMP для C.
Метод сеток хорошо подходит для распараллеливания.
Рассмотрим идею геометрического параллелизма.



Каждый процессор считает свой вертикальный фрагмент полосы, причем он должен получать и передавать информацию о посчитанных данных и их свойствах.
В данном методе ко времени собственно счета прибавляется время на передачу информации. Метод позволяет получать высокие ускорения, если все процессоры работают синхронно, с одинаковой скоростью.
Были рассмотрены два варианта распараллеливания. В обоих вариантов каждый слой разбивается на равные фрагменты, на которых вычисления будут проводиться параллельно. В первом, каждый процессор не переходит к следующему слою, пока на предыдущем не посчитаются все значения. Во втором, предполагается, что используется система с гомогенной топологией, и при переходе на следующий слой каждый процессор ожидает окончания вычислений только для своего фрагмента слоя.

Параллельный алгоритм I представлен в Listing 4, а параллельный алгоритм II в Listing 5.

```
Listing 4:
for (int k = 0; k < m - 1; k++)
        #pragma omp parallel for
        for (int i = 1; i < n; i++)
                u[i][k + 1] = koef1 * (u[i + 1][k] + u[i
                  - 1][k]) + koef2 * u[i][k];
```

```
Listing 5:
#pragma omp parallel for
for (int i = 0; i < nth; i++)
{
        int th = omp_get_thread_num();
        int p = n / nth;
        for (int k = 0; k < m - 1; k++)
                for (int i = th * p + 1; i < n && i < (th
                   + 1) * p; i++)
                        u[i][k + 1] = koef1 * (u[i + 1][k
                          ] + u[i - 1][k]) + koef2 * u[i
                          ][k];
}
```

В Listing 5 nth - число потоков.

6 Модельные задачи

Для анализа эффективности программ, написанных на C++, приведенных в Listing 1-5, были рассмотрены три модельные задачи.
Задача 1:

$$u_t - u_{xx} = 0,$$
$$u(x,0) = cos(0.5x) + (1-x)x,$$
$$u(0,t) = exp(t), u(1,t) = exp(-0.25t)cos(0.5).$$

Задача 2:

$$u_t - u_{xx} = 0,$$
$$u(x,0) = \begin{cases} x, x \in [0,1], \\ (2-x), x \in [1,2], \end{cases}$$
$$u(0,t) = u(2,t) = 0.$$

Аналитически полученное решение данной задачи:

$$u(x,t) = \frac{8}{\pi^2} \sum_{m=1}^{\infty} \frac{(-1)^{m-1}}{(2m-1)^2} (sin\frac{(2m-1)\pi x}{2}) e^{(\frac{(2m-1)\pi}{2})^2 t}, \tag{21}$$

Задача 3:

$$u_t - u_{xx} = 0, u(x,0) = 2sin(\pi x), u(0,t) = u(1,t) = 0.$$

Аналитически полученное решение данной задачи:

$$u(x,t) = (2sin(\pi x))e^{-\pi^2 t}$$

7 Методы, использованные для получения и анализа результатов

Необходимые вычисления проводились на четырехядерном процессоре Intel Core i7 3630QM с частотой 2.4 ГГц на операционной система Windows 10. Для измерения времени работы алгоритмов использовались средства библиотеки OpenMP(Listing 6).

```
Listing 6:
start_time = omp_get_wtime();
//algorithm
cout << "time of algorithm " << omp_get_wtime() -
   start_time << endl;
```

С помощью пакета Maple возможно получение точных значений в узлах сеток для, задач, которые возможно решить аналитически.
В Listing 7 приведена в качестве примера реализация получения подобных значений для модельной задачи 2.

```
Listing 7:
u := unapply(8*(sum((-1)^(m-1)*sin((1/2*(2*m-1))*Pi*x)*
   exp(-((1/2*(2*m-1))*Pi)^2*t)/(2*m-1)^2, m = 1 .. 20))/
   Pi^2, x, t);
a := 0;
b := 2;
n := 100;
m := 500;
h := 0.2e-1;
tau := 0.2e-3;
T := m * tau;
uxt := Matrix(n + 1, m, datatype = double);
for i to n + 1 do
        if a + (i - 1) * h < 1
        then
```

```
                uxt[i, 1] := a + (i − 1) * h
        else
                uxt[i, 1] := 2 − a − (i − 1) * h
        end if
end do;
for i to n + 1 do
        x[i] := a + (i − 1) * h;
        for j from 2 to m do
                if x[i] = 0 or x[i] = 2
                then
                        uxt[i, j] = 0
                else
                        y[j] := (j −1 ) * tau;
                        uxt[i, j] := u(x[i], y[j])
                end if
        end do
end do;
surfdata(uxt, a .. b, 0 .. T, orientation = [235, 75, 0])
    ;
}
```

Для оценки погрешности использовались средства ввода/вывод языка C++ в файл(Listing 8) с последующим импортом и обработкой в Maple(Listing 9).

```
Listing 8:
ofstream out;
out.open("u.txt");
for (int j = 0; j < m; j++)
{
        for (int i = 0; i < n; i++)
        {
                out << u[i][j] << ' ';
```

```
        }
        out << u[n][j] << endl;
}
out.close();
}
```

Listing 9:

```
u := ImportMatrix("u.txt", delimiter = "␣", transpose =
   true);
e := abs(u-uxt);
surfdata(u, a .. b, 0 .. T, orientation = [235, 75, 0]);
surfdata(e, a .. b, 0 .. T, orientation = [235, 75, 0])
```

8 Результаты

8.1 Результаты первой модельной задачи

Рассмотрим время работы программы для первой задачи(Таблица 1). Была взята равномерная сетка, n = 100, m = 500, для t был выбран шаг, удовлетворяющий условию устойчивости, равный $\frac{h^2}{2}$, где h - шаг по x, $a = 0, b = 1, h = 0.01, T = 0.025, \tau = 0.00005$.

	Полиномиальный сплайн	Тригонометрический сплайн
Последовательный алгоритм	0.12	0.14
Параллельный алгоритм I	0.06	0.06
Параллельный алгоритм II	0.05	0.05

Таблица 1. Время вычисления задачи при последовательном и параллельных алгоритмах(для четырехядерного процессора).

8.2 Результаты второй модельной задачи

Была взята равномерная сетка. n = 100, m = 500, для t был выбран шаг, удовлетворяющий условию устойчивости, равный $\frac{h^2}{2}$, где h - шаг по x,
$a = 0, b = 2, h = 0.02, T = 0.1, \tau = 0.0002$.
На Рис. 1 приведена поверхность, построенная с помощью точных значений. На Рис. 2 и Рис. 4 поверхности, построенных с помощью значений полученных с использованием полиномиальной и тригонометрической аппроксимации решения соответственно. На Рис. 3 и Рис. 5 их погрешности соответственно.

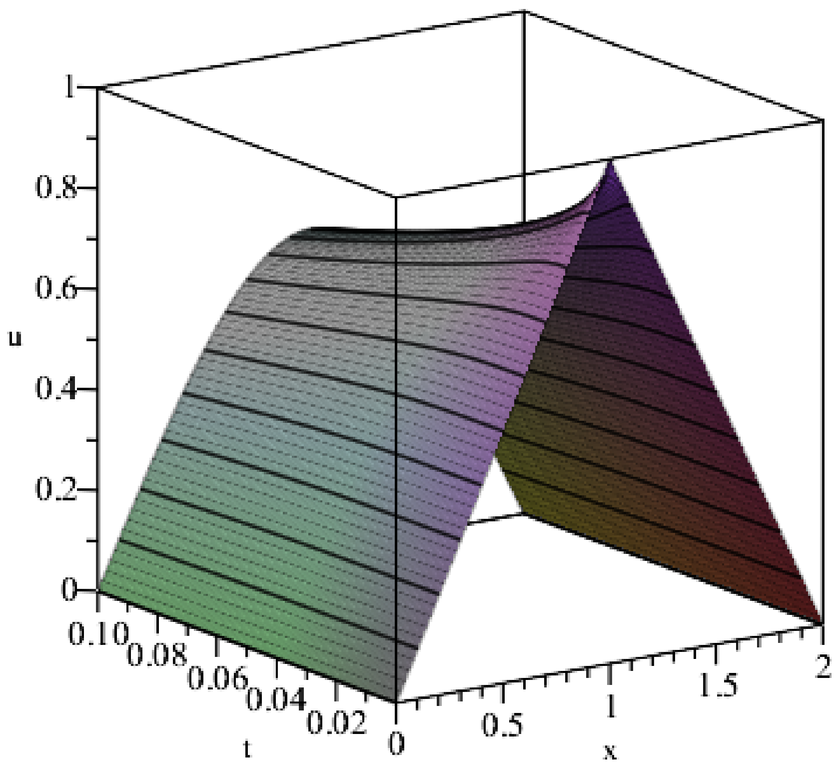


Рис. 1: Частичная сумма ряда (21) двадцати членов ряда

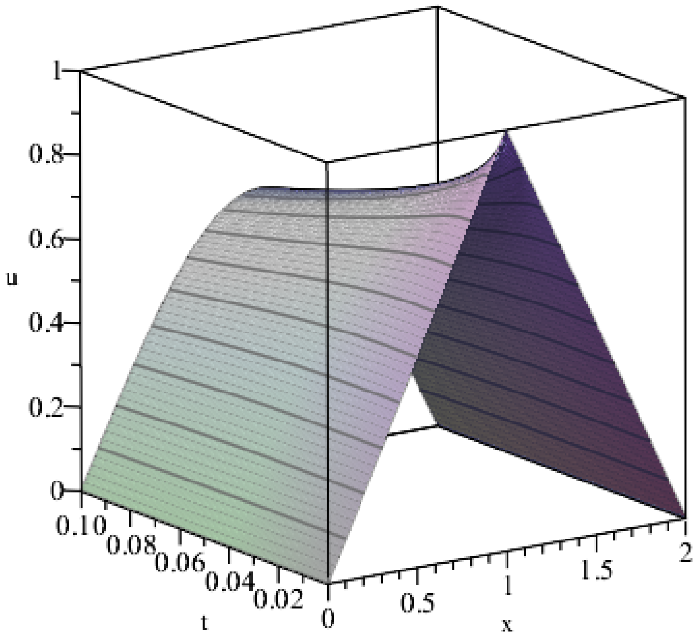


Рис. 2: Значения, полученные при использовании полиномиального сплайна

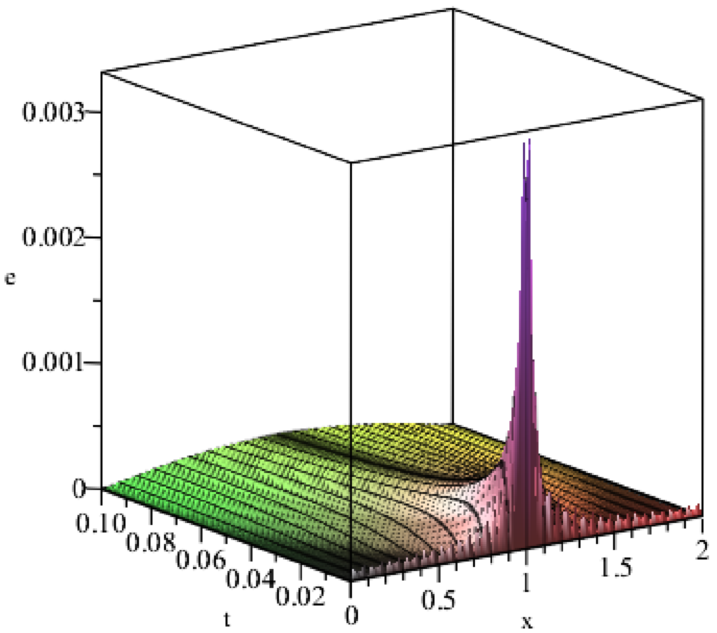


Рис. 3: Погрешность при использовании полиномиального сплайна

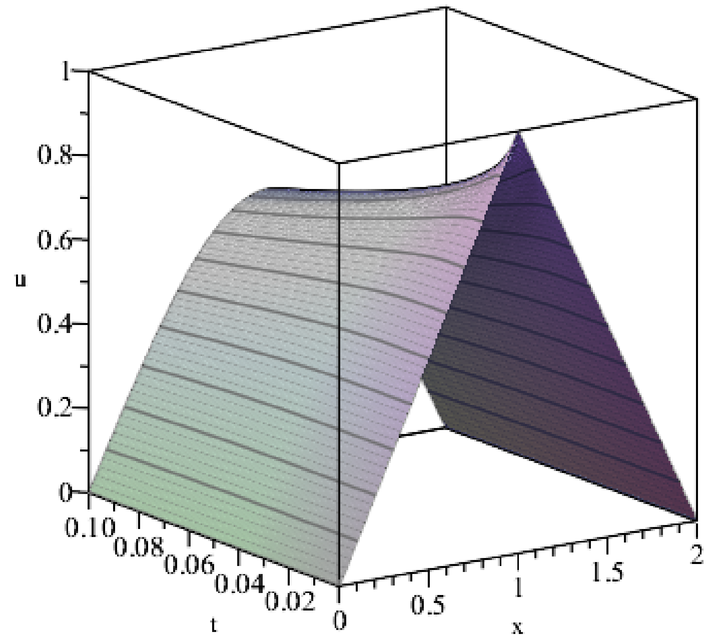


Рис. 4: Значения, полученные при использовании тригонометрического сплайна

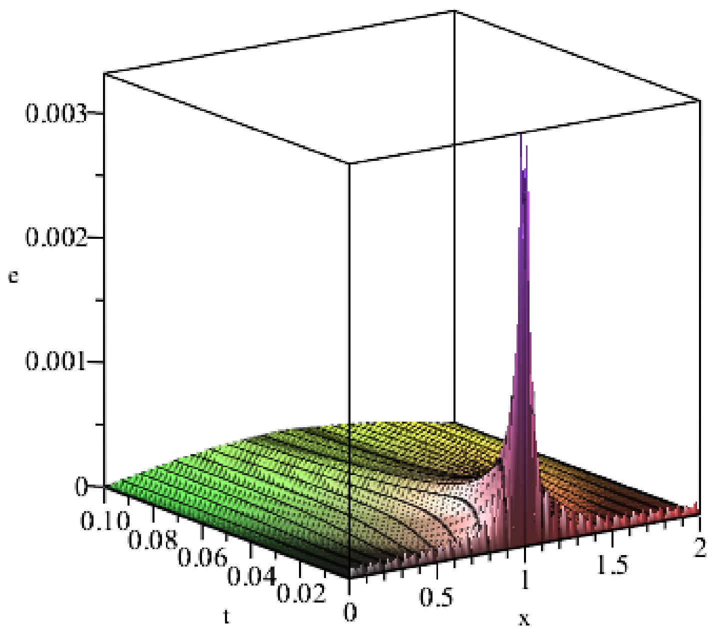


Рис. 5: Погрешность при использовании тригонометрического сплайна

Для исследования сходимости рассмотрим более мелкую сетку с шагом $\tau = 0.00002$, На на Рис. 6 и Рис. 8 поверхности, построенных с помощью значений полученных с использованием полиномиальной и тригонометрической аппроксимации соответственно на мелкой сетке, на Рис. 3 и Рис. 5 их погрешности соответственно.

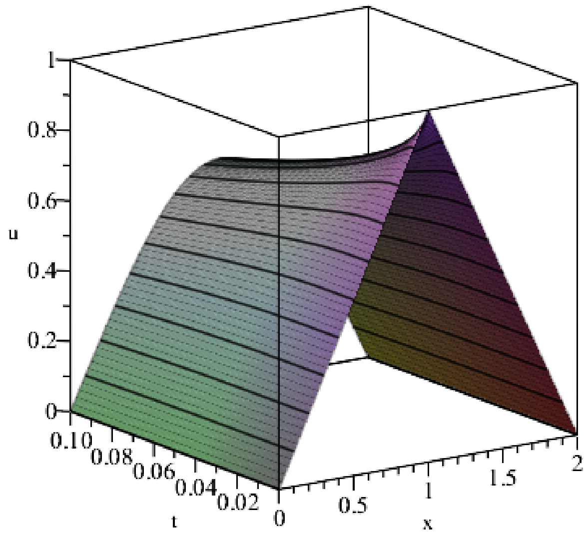


Рис. 6: Значения, полученные при использовании полиномиального сплайна, $\tau = 0.00002$

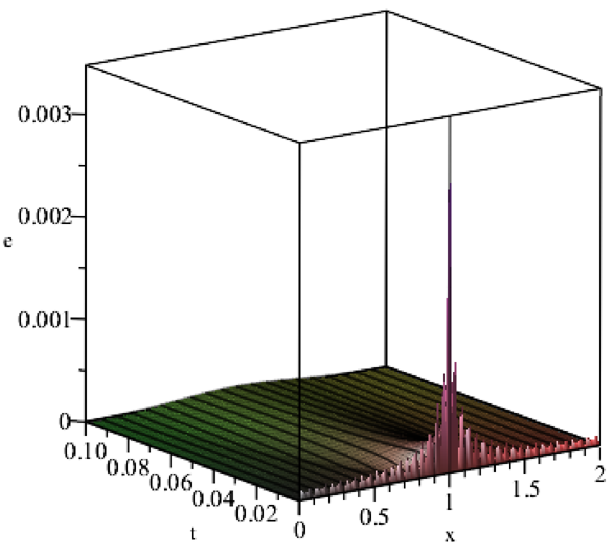


Рис. 7: Погрешность при использовании полиномиального сплайна,$\tau = 0.00002$

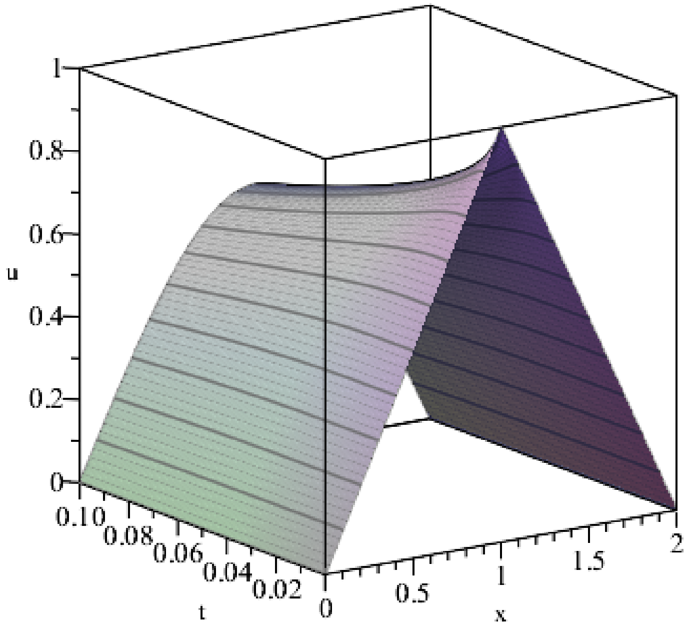


Рис. 8: Значения, полученные при использовании тригонометрического сплайна, $\tau = 0.00002$

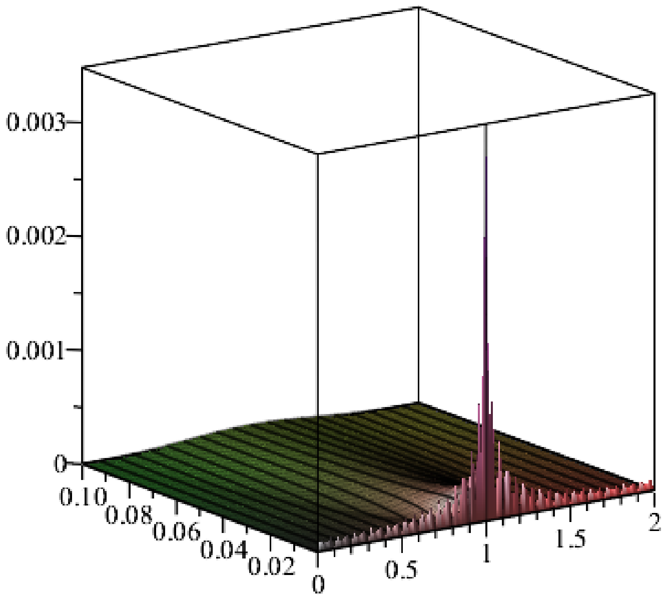


Рис. 9: Погрешность при использовании тригонометрического сплайна,$\tau = 0.00002$

Видно, что сходимость есть.

Рассмотрим поверхность разности между плоскостями погрешностей двух рассмотренных алгоритмов на большой сетке(Рис. 10 и 11) и на мелкой (Рис. 11 и 12).

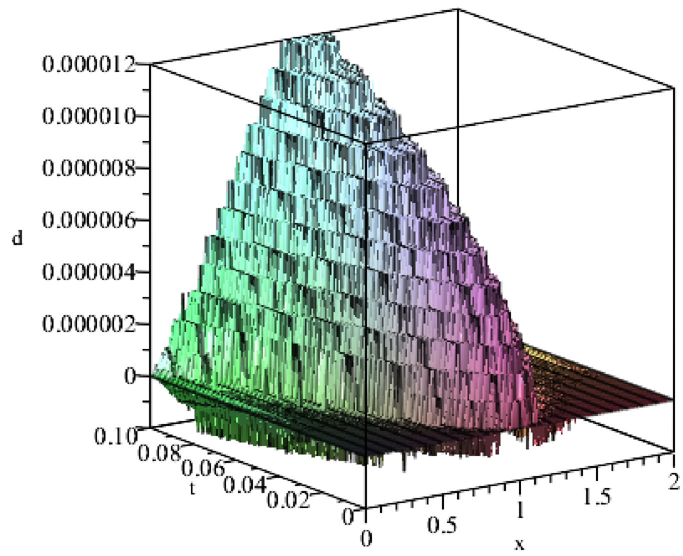


Рис. 10: Разность погрешностей алгоритмов с тригонометрическим и полиномиальным сплайнами

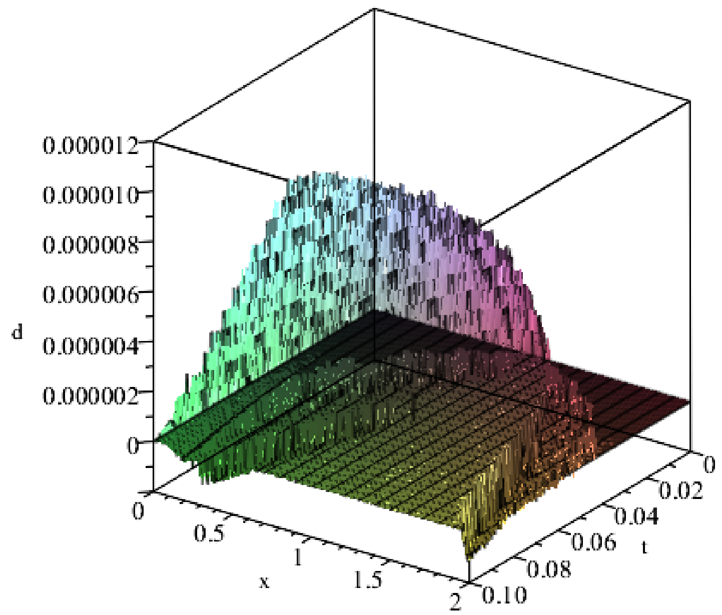


Рис. 11: Разность погрешностей алгоритмов с тригонометрическим и полиномиальным сплайнами

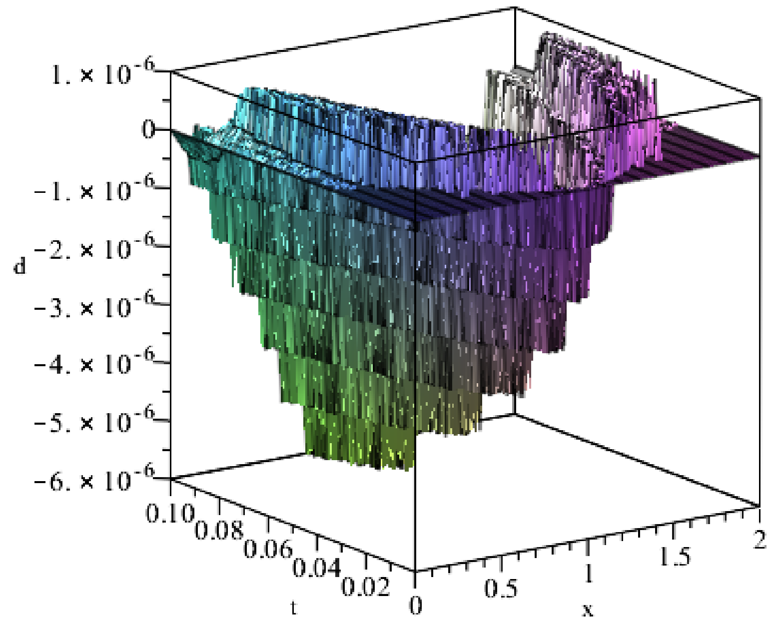


Рис. 12: Разность погрешностей алгоритмов с тригонометрическим и полиномиальным сплайнами, $\tau = 0.00002$

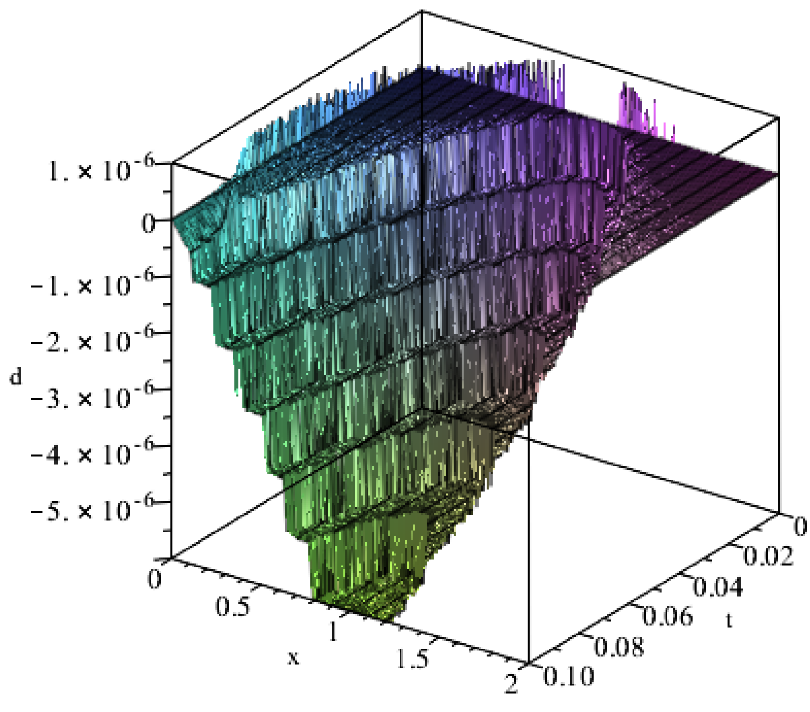


Рис. 13: Разность погрешностей алгоритмов с тригонометрическим и полиномиальным сплайнами, $\tau = 0.00002$

Видно, что в этой задаче вблизи границы тригонометрический сплайн дает меньшую погрешность, а внутри области большую по сравнению с полиномиальным. Однако, в случае более мелкой сетки тригонометрический сплайн дает более точную аппроксимацию решения.

Также рассмотрим графики значений погрешности на конкретных слоях как для полиномиального случая, так и для тригонометрического(Рис. 14-19).

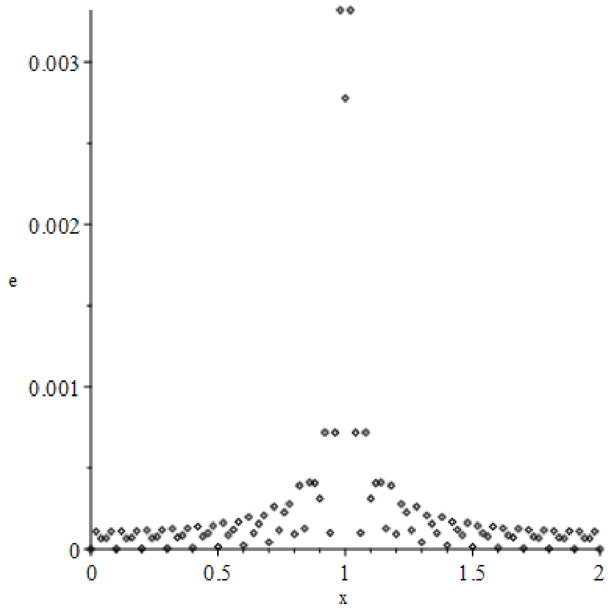


Рис. 14: График погрешности на первом слое при использовании полиномиального сплайна

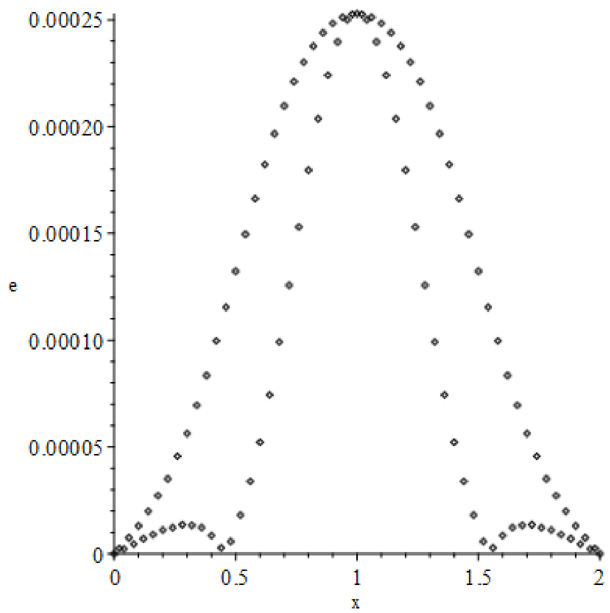


Рис. 15: График погрешности на 249-м слое при использовании полиномиального сплайна

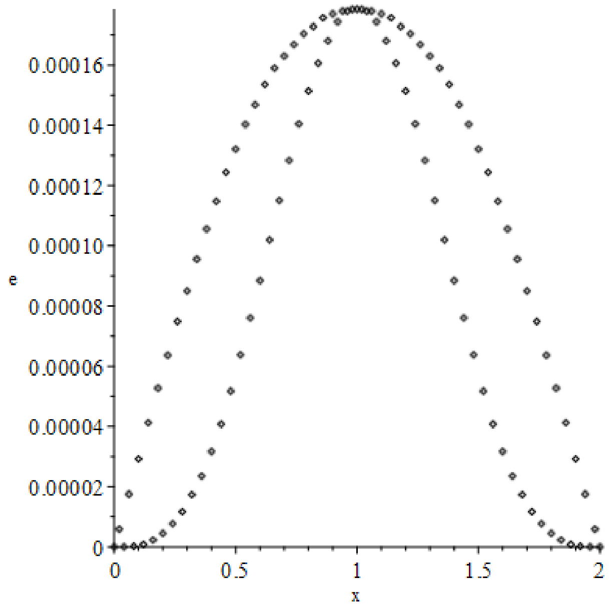


Рис. 16: График погрешности на 499-м слое при использовании полиномиального сплайна

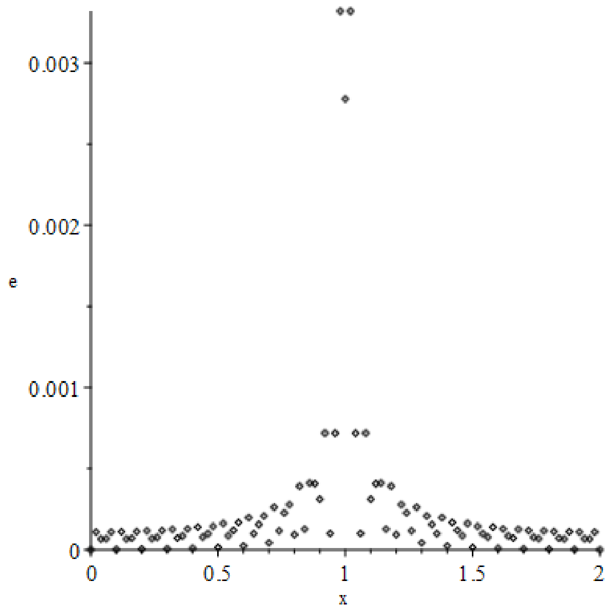


Рис. 17: График погрешности на первом слое при использовании тригонометрического сплайна

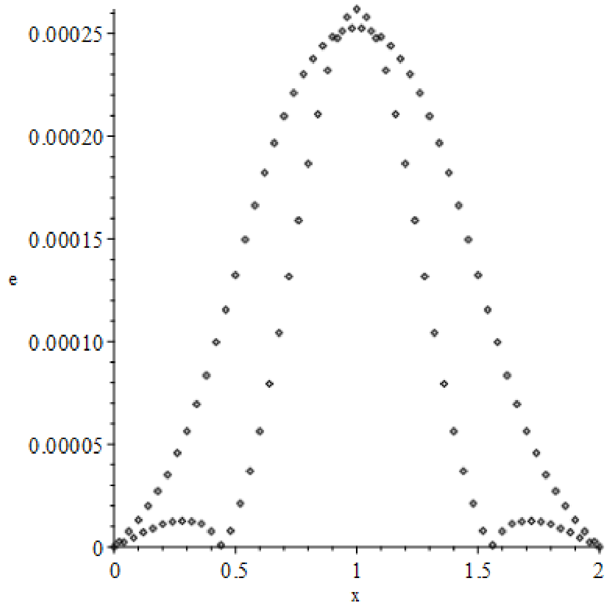


Рис. 18: График погрешности на 249-м слое при использовании тригонометрического сплайна

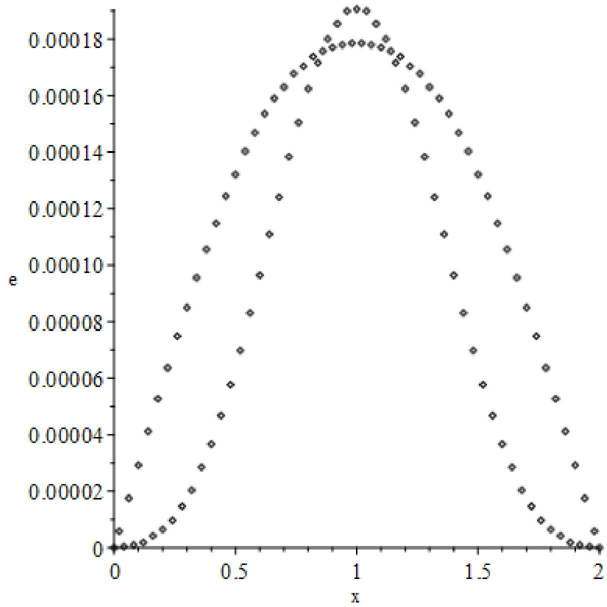


Рис. 19: График погрешности на 499-м слое при использовании тригонометрического сплайна

Рассмотрим время работы программы для второй задачи(Таблица 2).

	Полиномиальный сплайн	Тригонометрический сплайн
Последовательный алгоритм	0.12	0.14
Параллельный алгоритм I	0.05	0.06
Параллельный алгоритм II	0.04	0.05

Таблица 2. Время вычисления задачи при последовательном и параллельных алгоритмах(для четырехядерного процессора).

8.3 Результаты третьей модельной задачи

Была взята равномерная сетка. n = 100, m = 500, для t был выбран шаг, удовлетворяющий условию устойчивости, равный $\frac{h^2}{2}$, где h - шаг по x, $a = 0, b = 1, h = 0.01, T = 0.025, \tau = 0.00005$.

На Рис. 20 приведена поверхность, построенная с помощью точных значений. На Рис. 21 и Рис. 23 поверхности, построенных с помощью значений полученных с использованием полиномиальной и тригонометрической аппроксимации решения соответственно. На Рис. 22 и Рис. 24 их погрешности соответственно.

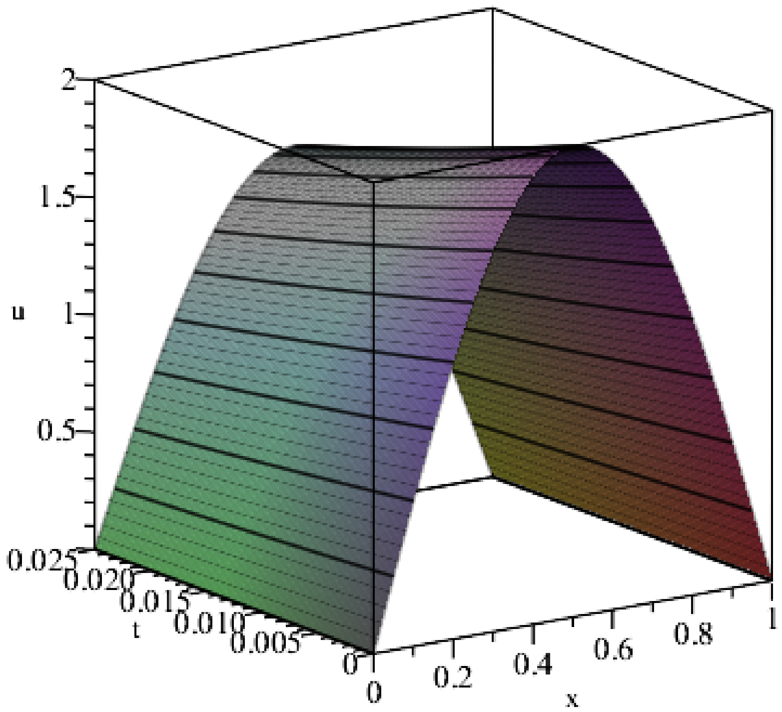


Рис. 20: Плоскость решения

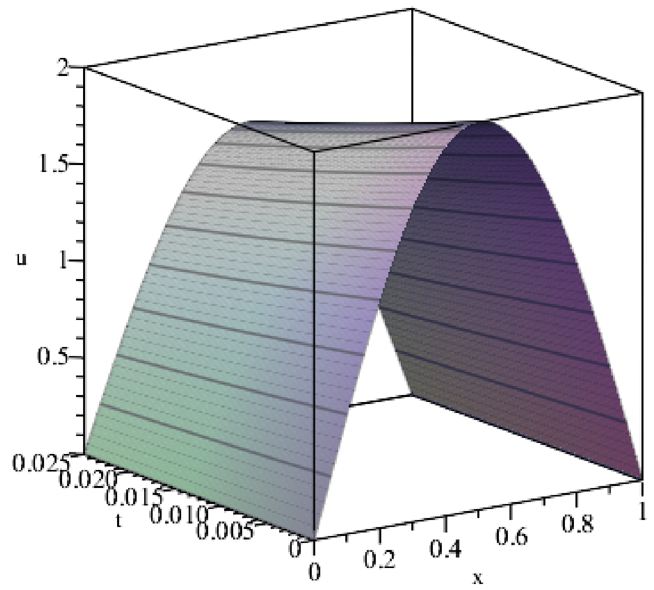


Рис. 21: Значения, полученные при использовании полиномиального сплайна

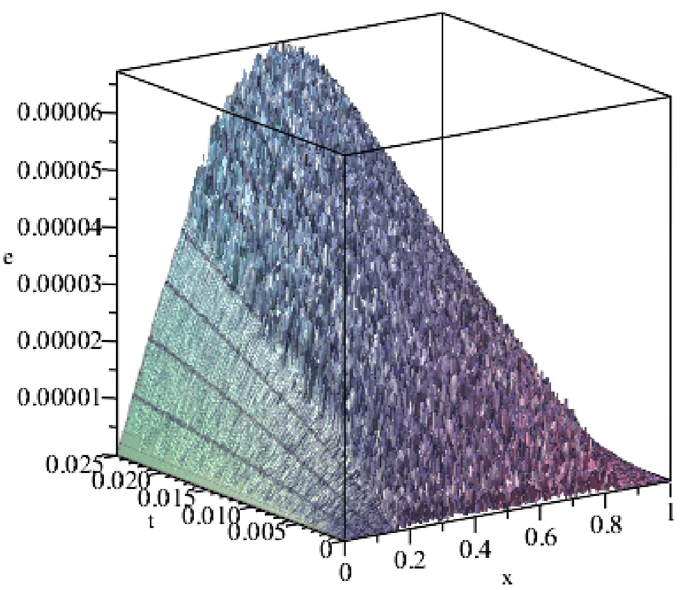


Рис. 22: Погрешность при использовании полиномиального сплайна

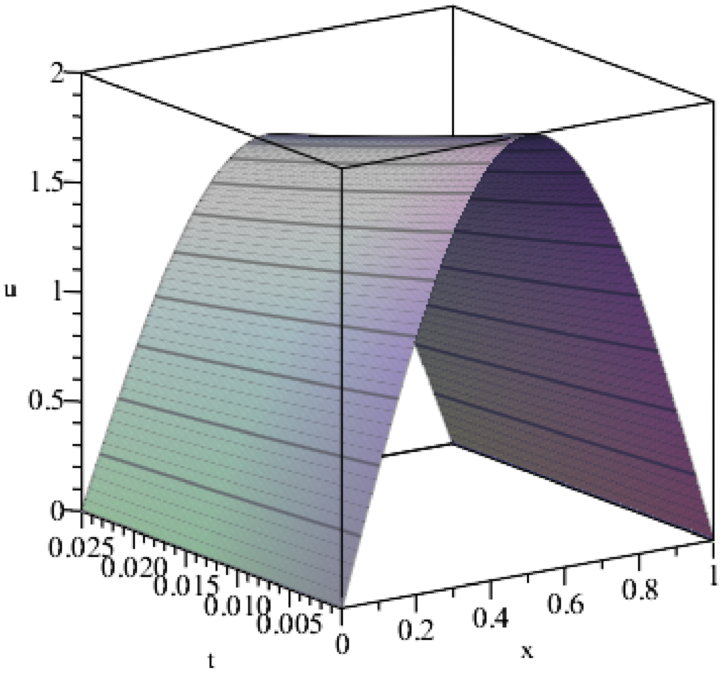


Рис. 23: Значения, полученные при использовании тригонометрического сплайна

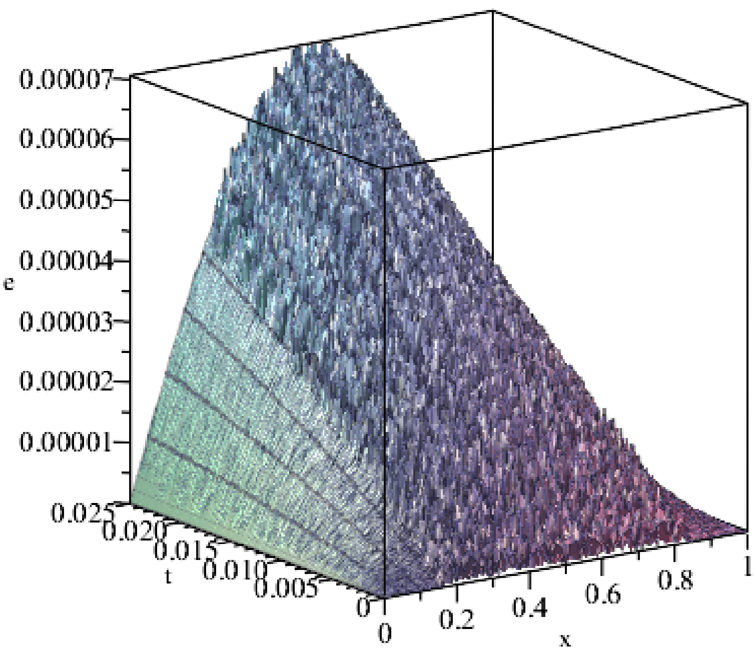


Рис. 24: Погрешность при использовании тригонометрического сплайна

Для исследования сходимости рассмотрим более мелкую сетку с шагом $\tau = 0.00002$, На на Рис. 25 и Рис. 27 поверхности, построенных с помощью значений полученных с использованием полиномиальной и тригонометрической аппроксимации соответственно на мелкой сетке, на Рис. 26 и Рис. 28 их погрешности соответственно.

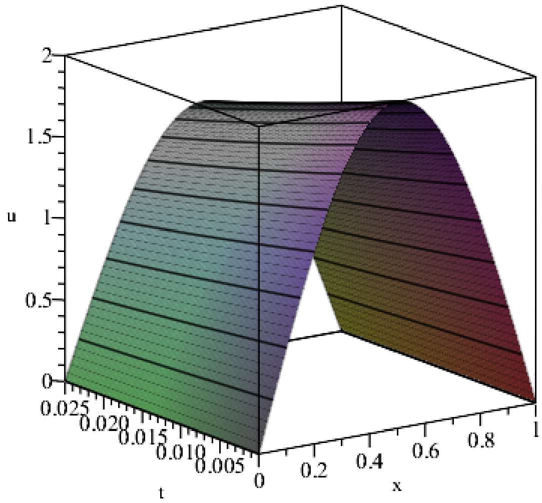


Рис. 25: Значения, полученные при использовании полиномиального сплайна, $\tau = 0.00002$

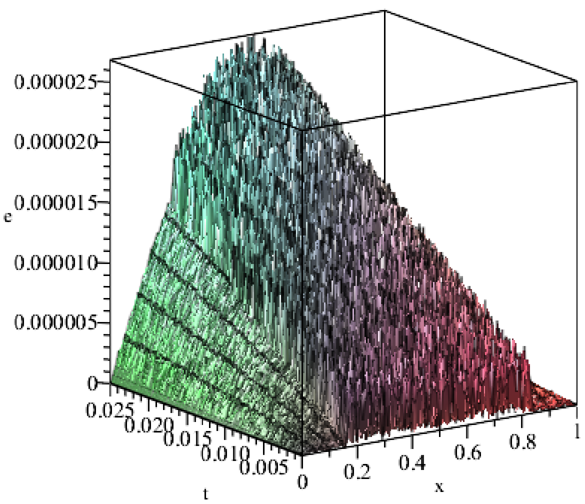


Рис. 26: Погрешность при использовании полиномиального сплайна, $\tau = 0.00002$

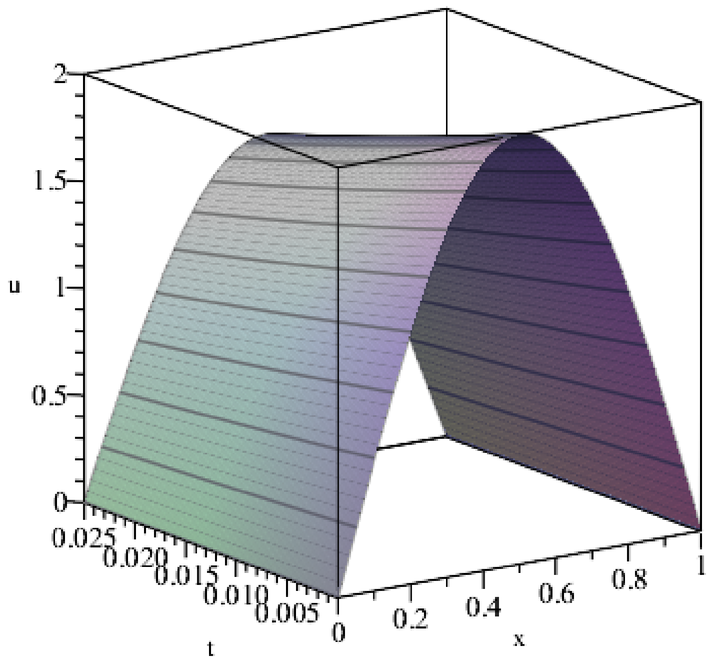


Рис. 27: Значения, полученные при использовании тригонометрического сплайна, $\tau = 0.00002$

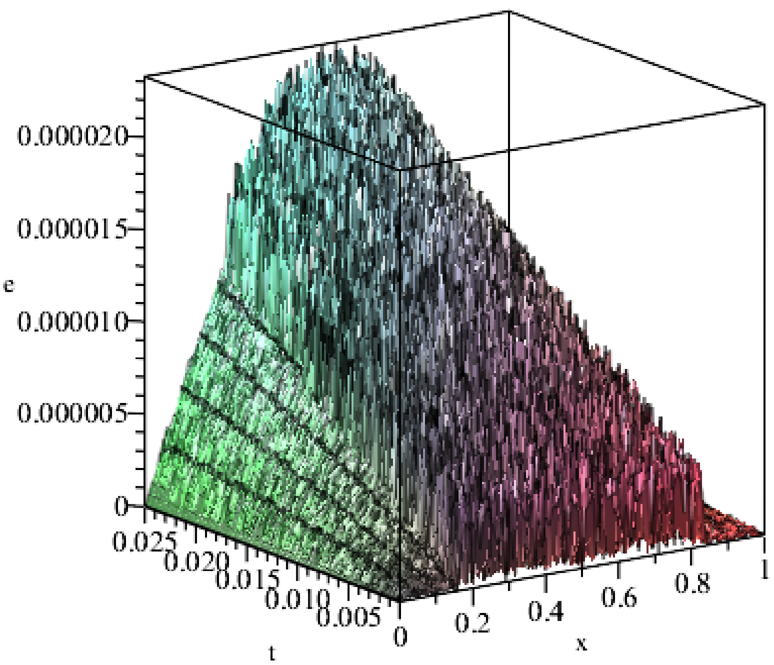


Рис. 28: Погрешность при использовании тригонометрического сплайна, $\tau = 0.00002$

Видно, что сходимость есть.

Рассмотрим поверхность разности между плоскостями погрешностей двух рассмотренных алгоритма на большой сетке(Рис. 29 и 31) и на мелкой (Рис. 30 и 32).

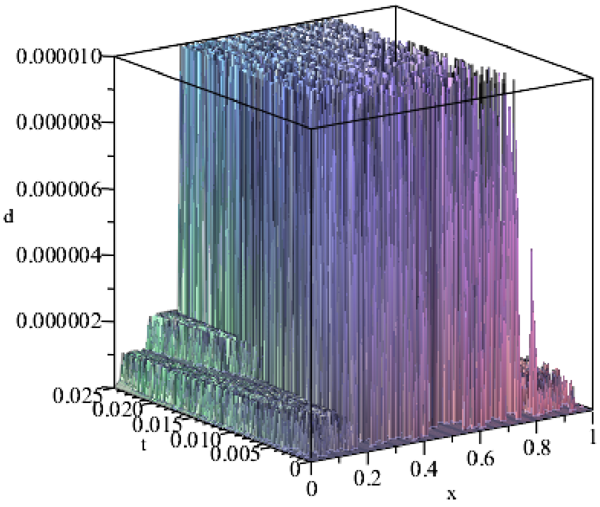


Рис. 29: Разность погрешностей алгоритмов с тригонометрическим и полиномиальным сплайнами

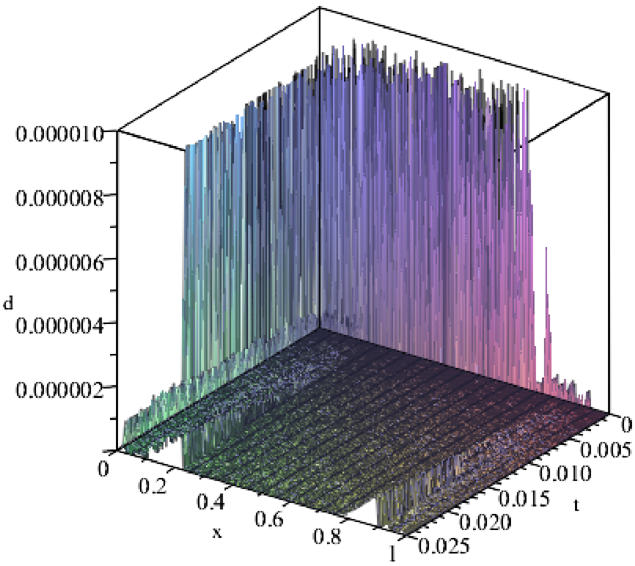


Рис. 30: Разность погрешностей алгоритмов с тригонометрическим и полиномиальным сплайнами

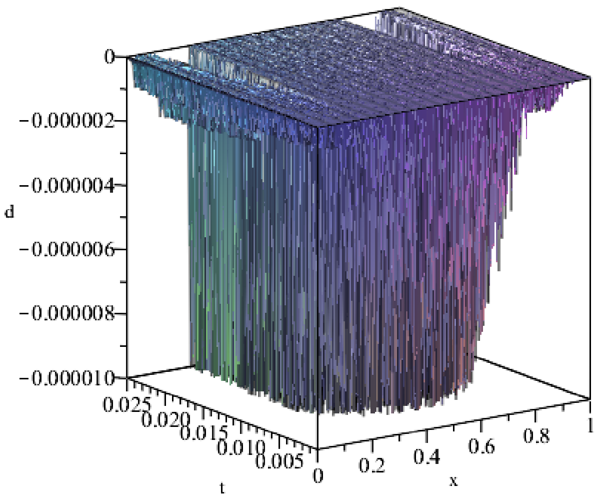


Рис. 31: Разность погрешностей алгоритмов с тригонометрическим и полиномиальным сплайнами, $\tau = 0.00002$

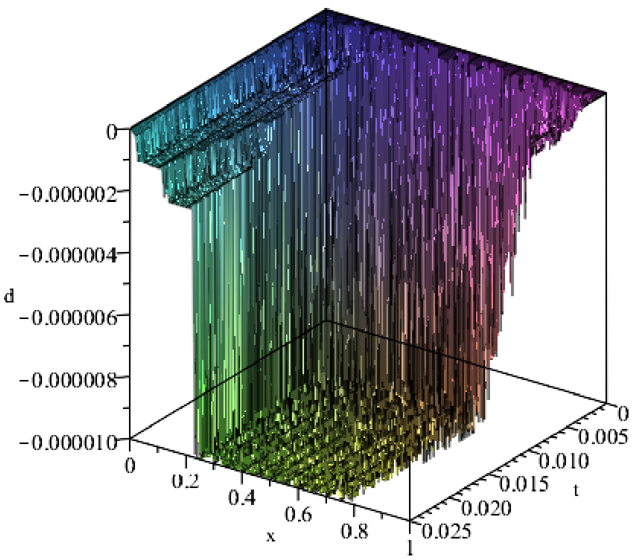


Рис. 32: Разность погрешностей алгоритмов с тригонометрическим и полиномиальным сплайнами, $\tau = 0.00002$

Видно, что в этой задаче вблизи границы тригонометрический сплайн дает меньшую погрешность, а внутри области большую по сравнению с полиномиальным. Однако, в случае более мелкой сетки тригонометрический сплайн дает более точную аппроксимацию решения. Также рассмотрим графики значений погрешности на конкретных слоях как для полиномиального случая, так и для тригонометрического(Рис. 32-37).

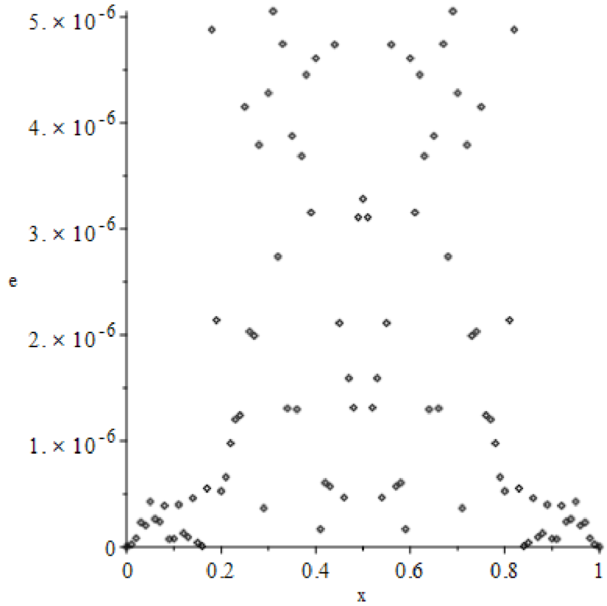


Рис. 33: График погрешности на первом слое при использовании полиномиального сплайна

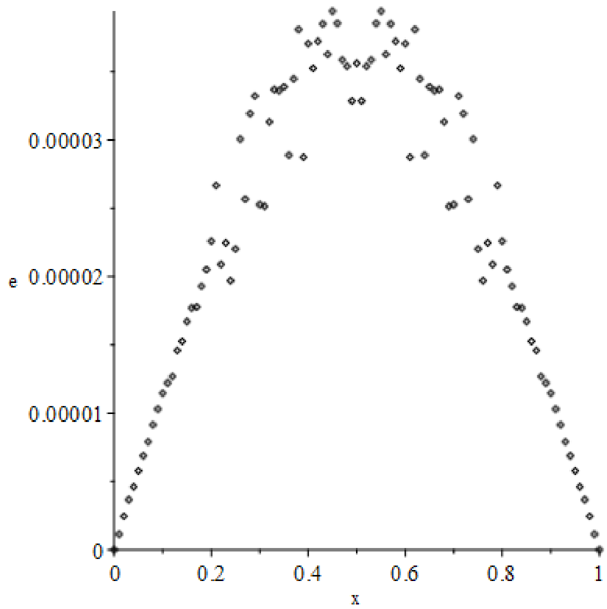


Рис. 34: График погрешности на 249-м слое при использовании полиномиального сплайна

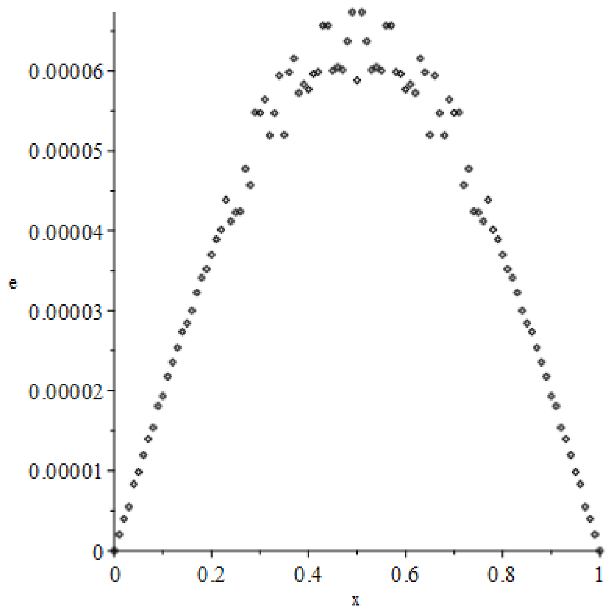


Рис. 35: График погрешности на 499-м слое при использовании полиномиального сплайна

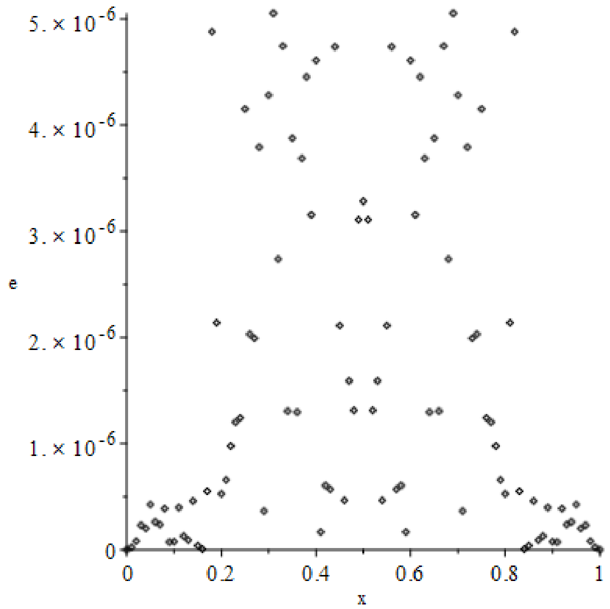


Рис. 36: График погрешности на первом слое при использовании тригонометрического сплайна

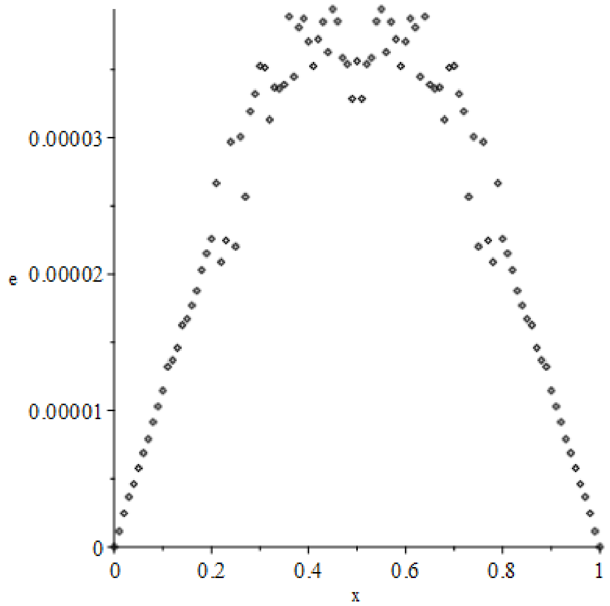


Рис. 37: График погрешности на 249-м слое при использовании тригонометрического сплайна

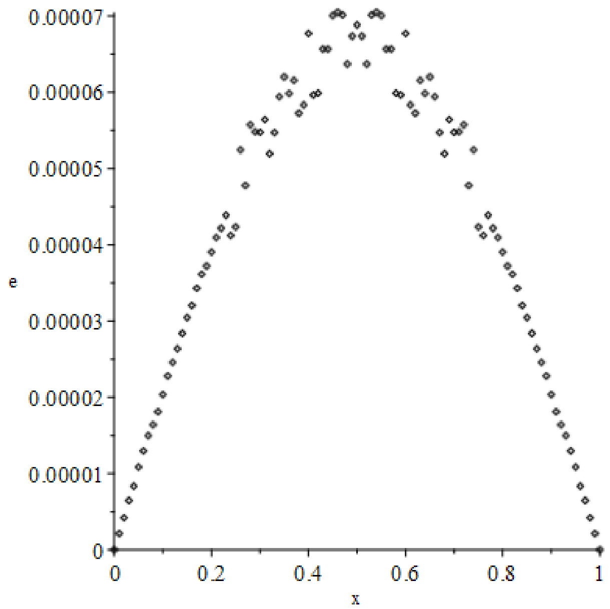


Рис. 38: График погрешности на 499-м слое при использовании тригонометрического сплайна

Рассмотрим время работы программы для второй задачи(Таблица 3).

	Полиномиальный сплайн	Тригонометрический сплайн
Последовательный алгоритм	0.12	0.12
Параллельный алгоритм I	0.05	0.05
Параллельный алгоритм II	0.04	0.04

Таблица 3. Время вычисления задачи при последовательном и параллельных алгоритмах(для четырехядерного процессора).

9 Заключение

В данной работе были рассмотрены полиномиальные квадратичные и тригонометрические сплайны, получены формулы для второй производной.

Была реализована два алгоритма явной схемы метода сеток в трех вариантах: последовательном и двух параллельных.

Были получены и проанализированы данные о времени работы алгоритмов, их сходимости и аппроксимации решения.

Из полученных результатов можно сделать вывод, что алгоритм с полиномиальной производной работает быстрее. Но ускорение при распараллеливании более эффективно в алгоритме с тригонометрическим сплайном.

Погрешность на сравнительно крупной сетке у полиномиального варианта меньше, но у тригонометрического алгоритма лучшая сходимость, и при измельчении сетки тригонометрический сплайн начинает лучше аппроксимировать решение. При этом, вблизи границ области сплайновая аппроксимация решения работает лучше по сравнению с полиномиальной, нежели в центре области.

Список источников

[1] Уравнения с частными производными : пер. с англ. / Эванс Л. К. ; ред. пер. Уральцева Н. Н. ; пер. Рожковская Т. Н. - Новосибирск : Тамара Рожковская, 2003. - XV, 560 с. - (Университетская серия ; т. 7). - Библиогр.: с. 557-560. - ISBN 5-901873-06-8.

[2] Ратыни А.К.: Уравнение теплопроводности. Иваново, 2007 г.

[3] Цирельман Н.М.: Теория и прикладные задачи тепломассопереноса. Уфа, 2002 г.

[4] Березин И.С., Жидков Н.П. Методы вычислений (том 2). М.: ГИФМЛ, 1959.

[5] Кузнецов Г.В., Шеремет М.А. Разностные методы решения задач теплопроводности. Томск: Издательство Томского Политехнического университета, 2007 г.

[6] Бурова И.Г, Демьянович Ю.К. Теория минимальных сплайнов. Изд-во С.-Пб. ун-та, 2000.

[7] Антонов А.С. Параллельное программирование с использованием технологии OpenMP. М.: Изд-во МГУ, 2009.