Московский авиационный институт
(Национальный исследовательский университет)

Институт: «Информационные технологии и прикладная математика»
Кафедра: 805 «Математическая кибернетика»
Дисциплина: «Численные методы»

Лабораторная работа №1
Тема: Решение начально-краевой задачи для дифференциального уравнения параболического типа

Студент:	Хахин Максим
Группа:	80-403
Преподаватель:	Иванов И. Э.
Дата:	
Оценка:	

Москва, 2021

1. **Задание:**
Используя явную и неявную конечно-разностные схемы, а также схему Кранка - Николсона, решить начально-краевую задачу для дифференциального уравнения параболического типа. Осуществить реализацию трех вариантов аппроксимации граничных условий, содержащих производные: двухточечная аппроксимация с первым порядком, трехточечная аппроксимация со вторым порядком, двухточечная аппроксимация со вторым порядком. В различные моменты времени вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением. Исследовать зависимость погрешности от сеточных параметров.

2. **Вариант 9:**
Значния $a = 5,\ b = 3,\ c = 0$

9.
$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial u}{\partial x},\ a > 0,\ b > 0.$$
$$u_x(0,t) - u(0,t) = -\exp(-at)(\cos(bt) + \sin(bt)),$$
$$u_x(\pi,t) - u(\pi,t) = \exp(-at)(\cos(bt) + \sin(bt)),$$
$$u(x,0) = \cos x,$$
Аналитическое решение: $U(x,t) = \exp(-at)\cos(x + bt)$.

3. **Теория:**

Разложение ДУ в ряд.

Диференциальное уравнение с частными производными параболического типа в общем виде записывается следующим образом:

$$\frac{\delta u}{\delta t} = a\frac{\delta^2 u}{\delta x^2} + b\frac{\delta u}{\delta x} + cu + f(x,t)$$

(a) Для явной разностной схемы.
Воспользуемся методами численного диференциирования и разложим каждый член уравнения имеющий производную в конечную сумму:

- $\frac{\delta u}{\delta t} = \frac{u_i^{k+1} - u_i^k}{\tau}$, где τ - шаг по временной сетке;
- $a\frac{\delta^2 u}{\delta x^2} = a\frac{u_{i-1}^k - 2u_i^k + u_{i+1}^k}{h^2}$, где h - шаг по пространственной сетке;
- $b\frac{\delta u}{\delta x} = b\frac{u_{i+1}^k - u_{i-1}^k}{2h}$;
- $cu = cu_i^k$;

Переппишем исходное уравнение:

$$\frac{u_i^{k+1} - u_i^k}{\tau} = a\frac{u_{i-1}^k - 2u_i^k + u_{i+1}^k}{h^2} + b\frac{u_{i+1}^k - u_{i-1}^k}{2h} + cu_i^k + f(x,t).$$

Сгруппируем коэффиценины при u с одинаковыми индексами:

$$u_i^{k+1} = \left(\frac{a}{h^2} - \frac{b}{2h}\right) u_{i-1}^k \tau + \left(c + \frac{1}{\tau} - \frac{2a}{h^2}\right) u_i^k \tau + \left(\frac{a}{h^2} + \frac{b}{2h}\right) u_{i+1}^k \tau + f(x,t)\tau.$$

Непосредственно из этого уравнения находится значение функции на временном уровне k+1.

(b) Для не явной разностной схемы.
Воспользуемся методами численного диференциирования и разложим каждый член уравнения имеющий производную в конечную сумму:

- $\frac{\delta u}{\delta t} = \frac{u_i^{k+1} - u_i^k}{\tau}$, где τ - шаг по временной сетке;
- $a\frac{\delta^2 u}{\delta x^2} = a\frac{u_{i-1}^{k+1} - 2u_i^{k+1} + u_{i+1}^{k+1}}{h^2}$, где h - шаг по пространственной сетке;
- $b\frac{\delta u}{\delta x} = b\frac{u_{i+1}^{k+1} - u_{i-1}^{k+1}}{2h}$;
- $cu = cu_i^{k+1}$;

Переппишем исходное уравнение:

$$\frac{u_i^{k+1} - u_i^k}{\tau} = a\frac{u_{i-1}^{k+1} - 2u_i^{k+1} + u_{i+1}^{k+1}}{h^2} + b\frac{u_{i+1}^{k+1} - u_{i-1}^{k+1}}{2h} + cu_i^{k+1} + f(x,t).$$

Сгруппируем коэффиценины при u с одинаковыми индексами:

$$u_i^k + f(x,t)\tau = \left(\frac{b}{2h} - \frac{a}{h^2}\right) u_{i-1}^{k+1} \tau + \left(\frac{1}{\tau} + \frac{2a}{h^2} - c\right) u_i^{k+1} \tau - \left(\frac{a}{h^2} + \frac{b}{2h}\right) u_{i+1}^{k+1} \tau.$$

Для нахождения значений функции u на уровне k+1 необходимо решить систему уравнеий:

$$\begin{cases} i = 1; \quad ... \\ i = \overline{2, N-1}; \quad Au_{i-1}^{k+1} + Bu_i^{k+1} + Cu_{i+1}^{k+1} = d_i \\ i = N; \quad ... \end{cases}$$

где $A = \left(\frac{b}{2h} - \frac{a}{h^2}\right)\tau$, $B = \left(\frac{1}{\tau} + \frac{2a}{h^2} - c\right)\tau$, $C = -\left(\frac{a}{h^2} + \frac{b}{2h}\right)\tau$, $d = u_i^k + f(x,t)\tau$.
Формирование уравнеий при i=1 и i=N рассмотрим в разделе посвященном краевые условия.

Примечание. *Во всех формулах приведенных выше рассматривается $i \in [2, N-1]$, где N - количекство узлов в сетке по координате x.*

Краевые условия.

Краевые условия могут принимать вид:

(a) Условия первого рода:

$$\begin{cases} \beta u(0,x) = \phi_l(t) \\ \delta u(l,x) = \phi_r(t) \end{cases}$$

В таком случае u_1^{k+1} и u_N^{k+1} находятся по формуле:

- Для явной разностной схемы
  $u_1^{k+1} = \frac{\phi_l(t)}{\beta}$, $u_N^{k+1} = \frac{\phi_r(t)}{\delta}$.
- Для не явной разностной схемы

$$\begin{cases} i = 1; \quad \beta u_1^{k+1} + 0 u_2^{k+1} = \phi_l(t) \\ i = \overline{2, N-1}; \quad A u_{i-1}^{k+1} + B u_i^{k+1} + C u_{i+1}^{k+1} = d_i \\ i = N; \quad 0 u_{N-1}^{k+1} + \delta u_N^{k+1} = \phi_r(t) \end{cases},$$

(b) Условия второго рода:

$$\begin{cases} \frac{\delta u}{\delta x}(0, x) = \frac{\phi_l(t)}{\alpha} \\ \frac{\delta u}{\delta x}(l, x) = \frac{\phi_r(t)}{\gamma} \end{cases} \Leftrightarrow \begin{cases} \frac{u_2^{k+1} - u_1^{k+1}}{h}(0, x) = \frac{\phi_l(t)}{\alpha} \\ \frac{u_N^{k+1} - u_{N-1}^{k+1}}{h}(l, x) = \frac{\phi_r(t)}{\gamma} \end{cases}$$

В таком случае u_1^{k+1} и u_N^{k+1} находятся по формуле:

- Для явной разностной схемы
  $u_1^{k+1} = u_2^{k+1} - h\frac{\phi_l(t)}{\alpha}$, $u_N^{k+1} = u_{N-1}^{k+1} + h\frac{\phi_r(t)}{\gamma}$.
- Для не явной разностной схемы

$$\begin{cases} i = 1; \quad -\frac{\alpha}{h} u_1^{k+1} + \frac{\alpha}{h} u_2^{k+1} = \phi_l(t) \\ i = \overline{2, N-1}; \quad A u_{i-1}^{k+1} + B u_i^{k+1} + C u_{i+1}^{k+1} = d_i \\ i = N; \quad -\frac{\gamma}{h} u_{N-1}^{k+1} + \frac{\gamma}{h} u_N^{k+1} = \phi_r(t) \end{cases},$$

(c) Условия третьего рода:

$$\begin{cases} \alpha \frac{\delta u}{\delta x}(0, x) + \beta u(0, x) = \phi_l(t) \\ \gamma \frac{\delta u}{\delta x}(l, x) + \delta u(l, x) = \phi_r(t) \end{cases} \Leftrightarrow \begin{cases} \alpha \frac{u_2^{k+1} - u_1^{k+1}}{h}(0, x) + \beta u_1^{k+1} = \phi_l(t) \\ \gamma \frac{u_N^{k+1} - u_{N-1}^{k+1}}{h}(l, x) + \delta u_N^{k+1} = \phi_r(t) \end{cases}$$

В таком случае u_1^{k+1} и u_N^{k+1} находятся по формуле:

- Для явной разностной схемы
  $u_1^{k+1} = \frac{\phi_l(t) - \frac{\alpha}{h} u_2^{k+1}}{\beta - \frac{\alpha}{h}}$, $u_N^{k+1} = \frac{\phi_r(t) + \frac{\gamma}{h} n_{N-1}^{k+1}}{\delta + \frac{\gamma}{h}}$.
- Для не явной разностной схемы

$$\begin{cases} i = 1; \quad \left(\beta - \frac{\alpha}{h}\right) u_1^{k+1} + \frac{\alpha}{h} u_2^{k+1} = \phi_l(t) \\ i = \overline{2, N-1}; \quad A u_{i-1}^{k+1} + B u_i^{k+1} + C u_{i+1}^{k+1} = d_i \\ i = N; \quad -\frac{\gamma}{h} u_{N-1}^{k+1} + \left(\delta + \frac{\gamma}{h}\right) u_N^{k+1} = \phi_r(t) \end{cases},$$

Примечание. *в разборе приведена апроксимация краевых условий с производной по 2 точкам с 1 порядком точности. Так же можно апроксимировать краевое условие по 3 точкам с точностью 2 и по 2 точкам с точностью 2.*

4. **Код:**

```
\\Define the initial boundary conditions
```

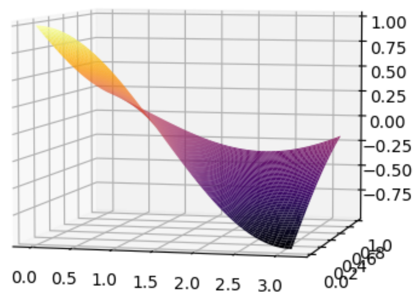
```
double phi(double x)
{
    return -exp(-1*x)*(cos(1*x)+sin(1*x));
}
double phil(double x)
{
    return exp(-1*x)*(cos(1*x)+sin(1*x));
}
double ksi(double x)
{
    return cos(x);
}
double f(double x, double t)
{
    return 0;
}


            \\explicit


void get_first_layer_explicit(int pr, int k, int N, double l,
 double T, double *args1, double *args2, matrix *grid,
  double(*phi)(double), double(*phil)(double), double(*ksi)(double),
   double(*f)(double, double))
{
    double tau = T/k, h = l/(N-1), mu = args1[1]*tau/(2*h);
    double b0, c0, bN, aN, d0, dN;

    for(int i=0; i<N; i++)
        *get_element(grid, 0, i) = ksi(i*h);

    for(int i=1; i<k; i++)
        for(int j=1; j<N-1; j++)
            *get_element(grid, i, j) = *get_element(grid, i-1, j+1)*
            (T + mu)+(*get_element(grid, i-1, j)*(1-2*T+ tau*args1[2]))+
            (*get_element(grid, i-1, j-1)*(T-mu));

    switch (pr)
    {
    case 1:
        for(int i=1; i<k; i++)
        {
            *get_element(grid, i, 0) = (*get_element(grid, i, 1)*
            (-args2[0]/h)+phi(tau*k))/(args2[1] - args2[0] / h);
            *get_element(grid, i, N-1) = (*get_element(grid, i, N-2)*
            (args2[2]/h)+phil(tau*k))/(args2[3] + args2[2] / h);
```

```
            }
            break;
        case 2:
            for(int i=1; i<k; i++)
            {
                *get_element(grid, i, 0) = ((*get_element(grid, i, 1)*
                4-*get_element(grid, i, 2))*(-args2[0]/(2*h))+phi(tau*k))/
                (args2[1] - 3*args2[0]/(2*h));
                *get_element(grid, i, N-1) = ((*get_element(grid, i, N-3)-
                *get_element(grid, i, N-2)*4)*(-args2[2]/(2*h))+phil(tau*k))/
                (args2[3] + 3*args2[2]/(2*h));
            }
            break;
        case 3:
            b0 = 2*args1[0]/h + h/tau - h*args1[2] - args2[1]/args2[0]*
            (2*args1[0] - args1[1]*h);
            c0 = -2*args1[0]/h;
            bN = 2*args1[0]/h + h/tau - h*args1[2] + args2[3]/args2[2]*
            (2*args1[0] + args1[1]*h);
            aN = -2*args1[0]/h;
            for(int i=1; i<k; i++)
            {
                d0 = h/tau * (*get_element(grid, i-1, 0)) - phi(tau*(k + 1)) *
                (2*args1[0] - args1[1]*h) / args2[0];
                dN = h/tau * (*get_element(grid, i-1, N-1))+ phil(tau*(k + 1)) *
                (2*args1[0] - args1[1]*h) / args2[2];
                *get_element(grid, i, 0) = (d0 - c0 *
                (*get_element(grid, i, 1))) / b0;
                *get_element(grid, i, N-1) = (dN - aN *
                (*get_element(grid, i, N-2))) / bN;
            }
            break;
    }
}


                \implicit


void get_first_layer_implicit(int pr, int k, int N, double l, double T,
 double *args1, double *args2, matrix *grid, double(*phi)(double),
  double(*phil)(double), double(*ksi)(double), double(*f)(double, double))
{
    double tau = T/k, h = l/(N-1);
    matrix system = create_matrix(N, N);
    matrix D = create_matrix(1,N);
```

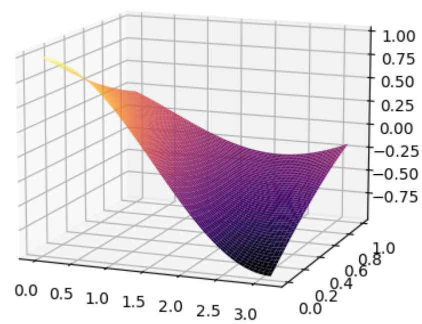
```
for(int i=0; i<N; i++)
    *get_element(grid, 0, i) = ksi(i*h);

for (int i=1; i<N-1; i++)
{
    *get_element(&system, i, i-1) = args1[0]*tau/pow(h,2)-
    args1[1]*tau/(2*h);
    *get_element(&system, i, i) = -2*args1[0]*tau/pow(h,2)+
    args1[2]*tau-1;
    *get_element(&system, i, i+1) = args1[0]*tau/pow(h,2)+
    args1[1]*tau/(2*h);
}

switch (pr)
{
case 1:
    *get_element(&system, 0, 0) = args2[1] - args2[0] / h;
    *get_element(&system, 0, 1) = args2[0] / h;
    *get_element(&system, N-1, N-2) = - args2[2] / h;
    *get_element(&system, N-1, N-1) = args2[3] + args2[2] / h;
    break;
case 2:
    *get_element(&system, 0, 0) = args2[1] - 3*args2[0]/ h/ 2;
    *get_element(&system, 0, 1) = 2 * args2[0]/h;
    *get_element(&system, 0, 2) = - args2[0] / h /2;
    *get_element(&system, N-1, N-3) = args2[2] / h /2;
    *get_element(&system, N-1, N-2) = -2 * args2[2]/h;
    *get_element(&system, N-1, N-1) = args2[3] + 3*args2[2]/ h/ 2;
    break;
case 3:
    *get_element(&system, 0, 0) = 2 * args1[0] / h + h / tau - h *
    args1[2] - args2[1] / args2[0] * (2 * args1[0] - args1[1] * h);
    *get_element(&system, 0, 1) = -2 * args1[0] / h;
    *get_element(&system, N-1, N-2) = -2 * args1[0] / h;
    *get_element(&system, N-1, N-1) = 2 * args1[0] / h + h / tau -
    h * args1[2] + args2[3] / args2[2] * (2 * args1[0] + args1[1] * h);
    break;
}
for(int i=1; i<k; i++)
{
    for(int j=1; j<N-1; j++)
        *get_element(&D, 0, j) = -(*get_element(grid, i-1, j));
    switch (pr)
    {
    case 1:
        *get_element(&D, 0, 0) = phi(i*tau);
        *get_element(&D, 0, N-1) = phil(i*tau);
```

```
                insert_matrix_line(grid, run_method(system, D, N), i);
                break;
            case 2:
                *get_element(&D, 0, 0) = phi(i*tau);
                *get_element(&D, 0, N-1) = phil(i*tau);
                insert_matrix_line(grid, LU_method(system, D, N), i);
                break;
            case 3:
                *get_element(&D, 0, 0) = h / tau * (*get_element(grid, i-1, 0))-
                phi(tau*i) * (2 * args1[0] - args1[1] * h) / args2[0];
                *get_element(&D, 0, N-1) = h/tau*(*get_element(grid, i-1, N-1))+
                phil(tau*i) * (2 * args1[0] + args1[1] * h) / args2[2];
                insert_matrix_line(grid, run_method(system, D, N), i);
                break;
        }
    }
    //destroy_matrix(&system); destroy_matrix(&D); destroy_matrix(&tmp);
}


                \\CrankNicolson


void get_first_layer_CrankNicolson(int pr, int k, int N, double sgm,
double l, double T,double *args1, double *args2, matrix *grid,
double(*phi)(double), double(*phil)(double),
double(*ksi)(double), double(*f)(double, double))
{
    matrix grid1 = create_matrix(k,N);
    matrix grid2 = create_matrix(k,N);
    get_first_layer_explicit(pr, k, N, l, T, args1, args2,
    &grid1, phi, phil, ksi, f);
    get_first_layer_implicit(pr, k, N, l, T, args1, args2,
    &grid2, phi, phil, ksi, f);
    printf("y");
    for (int i=0; i<k; i++)
        for (int j=0; j<N; j++)
            *get_element(grid, i, j) = *get_element(&grid1, i, j)*(1-sgm)+
            *get_element(&grid2, i, j)*sgm;
}
```

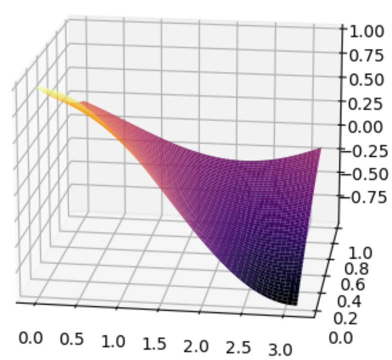
5. **Результат:**
   Явный метод сетка 1000*100:



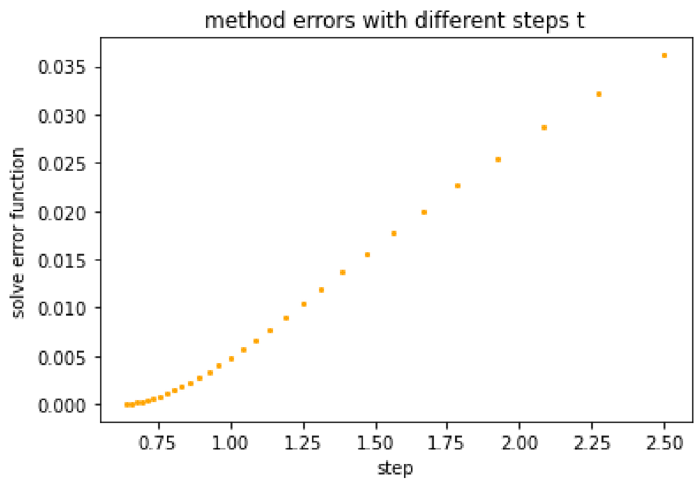
Ошибка : 0.0857908889653587
Явный метод сетка 2000*200:



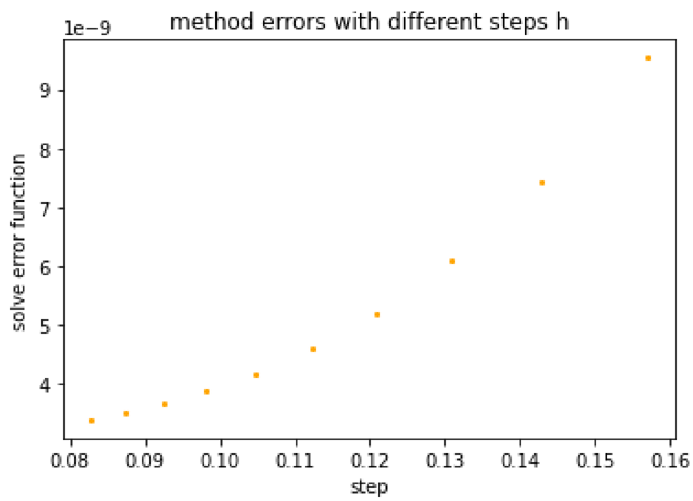
Ошибка: 0.0792584889653587
Явный метод сетка 4000*400:



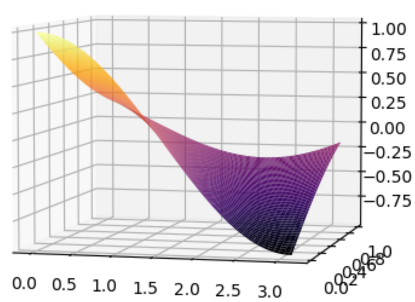
Ошибка : 0.0565016889653587
Ошибка с разным шагом по времени:



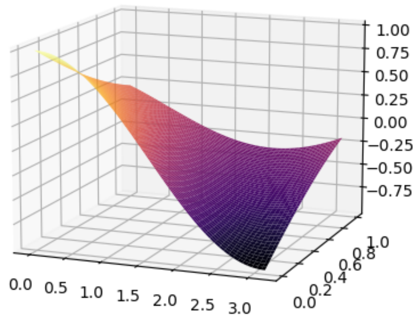
Ошибка с разным шагом по пространству:



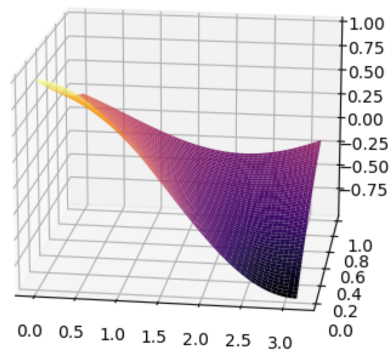
Неявный метод сетка 1000*100:



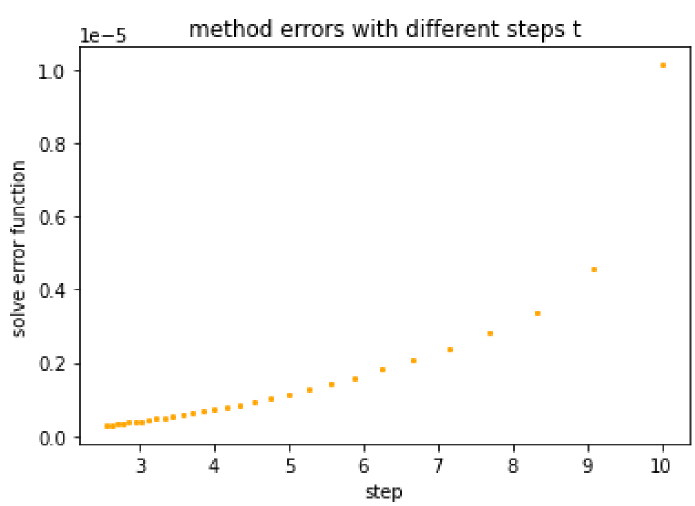
Ошибка : 0.007134110346412981
Неявный метод сетка 2000*200:



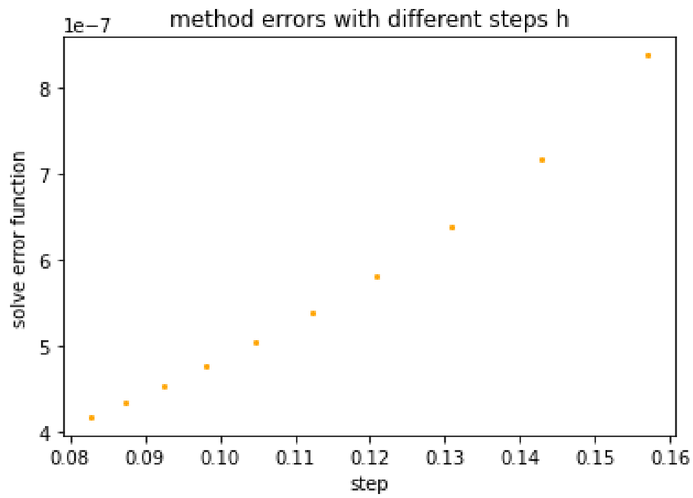
Ошибка: 0.0070211103464130065
Неявный метод сетка 4000*400:



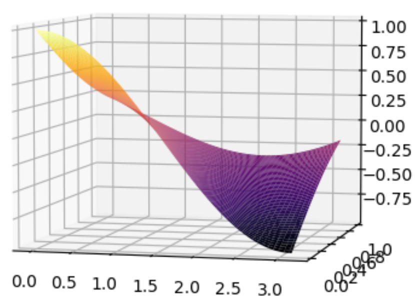
Ошибка : 0.006996110346412981 5
Ошибка с разным шагом по времени:



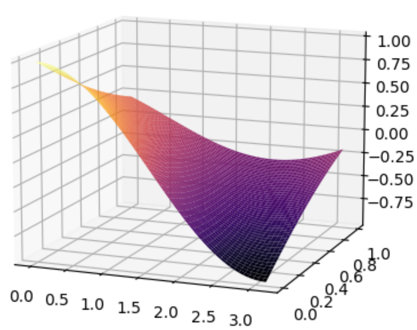
Ошибка с разным шагом по пространству:



КН метод сетка 1000*100:

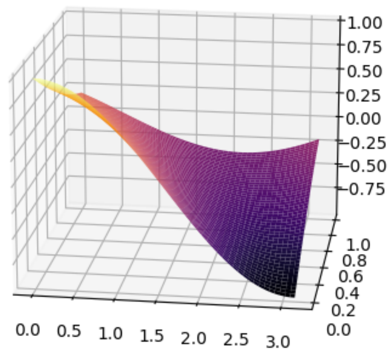


Ошибка : 0.0000222043352098354
КН метод сетка 2000*200:

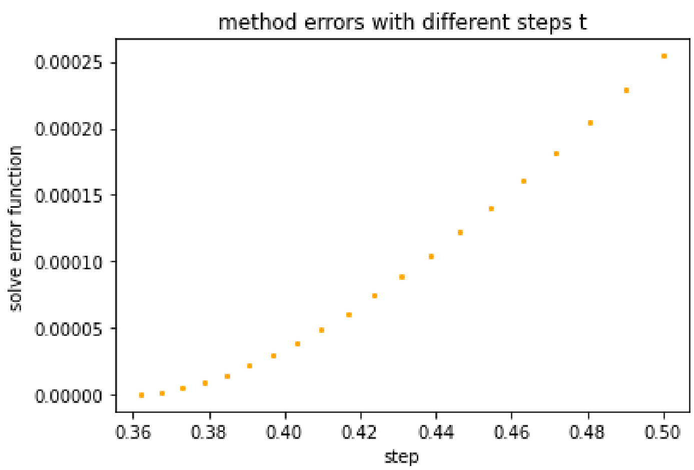


Ошибка: 0.00001963083520983543

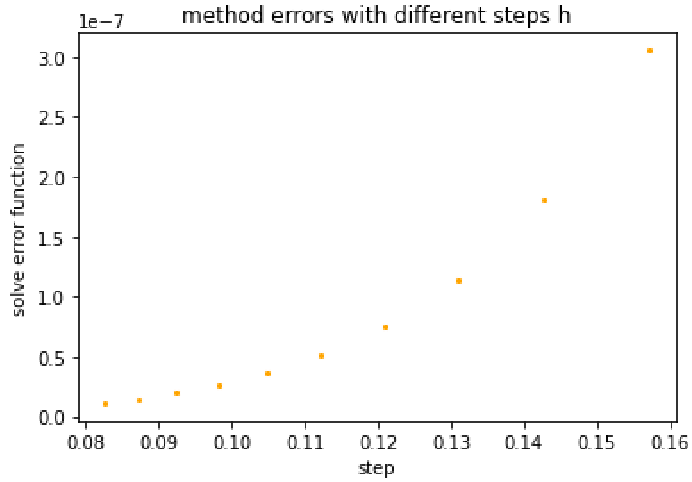
КН метод сетка 4000*400:



Ошибка : 0.000011319135209835429
Ошибка с разным шагом по времени:



Ошибка с разным шагом по пространству:



6. **Вывод:**

   В результате выполнения лабораторной работы были освоены 3 конечно-разностные схемы для решения начально-краевой задачи для дифференциального уравнения параболического типа : явная конечно-разностная схема, неявная конечно-разностная схема и схема Кранка - Николсона.

   Были построены графики ошибок, на которых показано падение ошибки при росте числа разбиений по пространству и времени.