Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования "Уфимский государственный авиационный технический университет"

Кафедра Высокопроизводительных вычислительных технологий и систем

Дисциплина: Теория разностных схем.

Отчет по лабораторной работе № 2

Тема: «Решение начально-краевой задачи для уравнения теплопроводности»

Группа ПМ-353	Фамилия И.О.	Подпись	Дата	Оценка
Студент	Шамаев И.Р.			
Принял	Белевцов Н.С.			

Уфа 2022

Цель работы: получить навык численного решения линейных и нелинейных начально-краевых задач для уравнений параболического типа с использованием различных конечно-разностных схем на примере задачи для одномерного уравнения теплопроводности с источником.

Теоретический материал

Начально-краевая задача

Рассматривается начально-краевая задача для нелинейного одномерного уравнения теплопроводности с источником:

$$\frac{\partial u}{\partial t}=\frac{\partial}{\partial x}\left(k(u)\frac{\partial u}{\partial x}\right)+f(t,x,u), x\in(0,1), t>0; \quad (1)$$

$$u(0,x)=\varphi(x), x\in[0,1]; \quad (2)$$

$$\alpha_0 u(t,0)+\beta_0 u_x(t,0)=\psi_0(t), t>0; \quad (3)$$

$$\alpha_1 u(t,1)+\beta_1 u_x(t,1)=\psi_1(t), t>0. \quad (4)$$

I. Сравнение конечно-разностных схем для линейной задачи

Рассматривается линейный случай уравнения теплопроводности (1):

$$k(u)=k_0=const, f(t,x,u)=f(t,x). \quad (5)$$

Параметры задачи выбираются в соответствии с индивидуальным заданием (Таблица 1).

Методом разделения переменных построить аналитическое решение задачи.

Задача 1. Явная разностная схема.

При решении данной задачи (1)-(5) используется явная конечно-разностная схема с шаблоном «левый уголок» на равномерной пространственно-временной сетке:

$$\frac{y_i^{n+1}-y_i^n}{\tau}=a\frac{y_{i+1}^n-2y_i^n+y_{i-1}^n}{h^2}+f(x_i,t^n).$$

Порядок аппроксимации данной схемы: $O(\tau+h^2)$.

Явная схема является условно устойчивой с условием

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

Задача 2. Неявная разностная схема.

При решении данной задачи (1)-(5) используются неявная конечно-разностная схема и схема Кранка-Николсона.

Неявная схема представляется следующим образом:

$\frac{y_i^{n+1}-y_i^n}{\tau}=a\frac{y_{i-1}^{n+1}-2y_i^{n+1}+y_{i+1}^{n+1}}{h^2}+f\left(x_i,t^{n+1}\right)$;Схема Кранка-Схема Кранка-Николсона имеет вид:

$$\frac{y_i^{n+1}-y_i^n}{\tau}=\frac{a}{h^2}\left[\left(u_{i-1}^{n+1}-2u_i^{n+1}+u_{i+1}^{n+1}\right)+(1-\gamma)\left(u_{i-1}^n-2u_i^n+u_{i+1}^n\right)\right]+f\left(x_i,t^{n+1}\right);$$

Порядок аппроксимации полностью неявной схемы: $O\left(\tau+h^2\right)$, а схемы Кранка-Николсона - $O\left(\tau^2+h^2\right)$. Полностью неявная схема устойчива при любых соотношениях между шагами сетки $\tau u h$, схема Кранка-Николсона обычно абсолютно устойчива, $\gamma=\frac{1}{2},\tau$ выбирается как $\tau\le\frac{h^2}{(2-4\gamma)}$.

II. Решение нелинейной задачи с использованием консервативной схемы.

Решается нелинейная задача (1)-(4) с дополнительными исходными данными $k(u)$ и $F(u)$из таблицы 2, где

$$f(t,x,u)=F(u)f(t,x),$$

а функция $f(t,x)$ и остальные данные берутся из таблицы 1.

Задача 3. Консервативная схема на равномерной сетке.

При решении данной задачи (1)-(4) используется консервативная схема на равномерной сетке:

$$y_j^{n+1}\left(\frac{1}{\tau}+\frac{\sigma}{h^2}v_{j+\frac{1}{2}}^{n+1}+\frac{\sigma}{h^2}v_{j-\frac{1}{2}}^{n+1}\right)-\left(\frac{\sigma}{h^2}v_{j+\frac{1}{2}}^{n+1}\right)y_{j+1}^{n+1}-\left(\frac{\sigma}{h^2}v_{j-\frac{1}{2}}^{n+1}\right)y_{j-1}^{n+1}=\left(\frac{1}{\tau}-\frac{(1-\sigma)}{h^2}v_{j+\frac{1}{2}}^n-\frac{(1-\sigma)}{h^2}v_{j-\frac{1}{2}}^n\right)y_j^n+\frac{1-\sigma}{h^2}\left(v_{j+\frac{1}{2}}^n\right)u_{j+1}^n-$$

где $v_{j+\frac{1}{2}}^n=K\left((v¿¿j^n+v_{j+1}^n)/2\right),v_{j-\frac{1}{2}}^n=K¿¿$

k_0	$f(t,x)$	α_0	β_0	α_1	β_1	$\varphi(x)$	$\psi_0(x)$	$\psi_1(x)$
1	$e^{-t}x$	1	-1	1	0	$\frac{\sin x+\cos x}{\sin 1}-x$	e^{-t}	$e^{-t}ctg\,1$

Задача 1 (2 балла).

1) Написать вычислительную программу на языке программирования C++ решения задачи с использованием явной разностной схемы на равномерной пространственно-временной сетке.
2) Непосредственными расчетами продемонстрировать условную устойчивость схемы и справедливость условия устойчивости.
3) Исследовать зависимость решения от величины шагов сетки по пространственной и временной переменным посредством сравнения с построенным аналитическим решением. Построить графики зависимости погрешности, оцениваемой в равномерной норме по пространственной переменной, от времени и шагов сетки.

Аналитическое решение:

$$u = e^{-t}\left[\frac{\sin x+\cos x}{\sin\ 1} - x\right]$$

```
N = 500, tau = 2e-06

time: 0.013
Error: 0.000941666
```

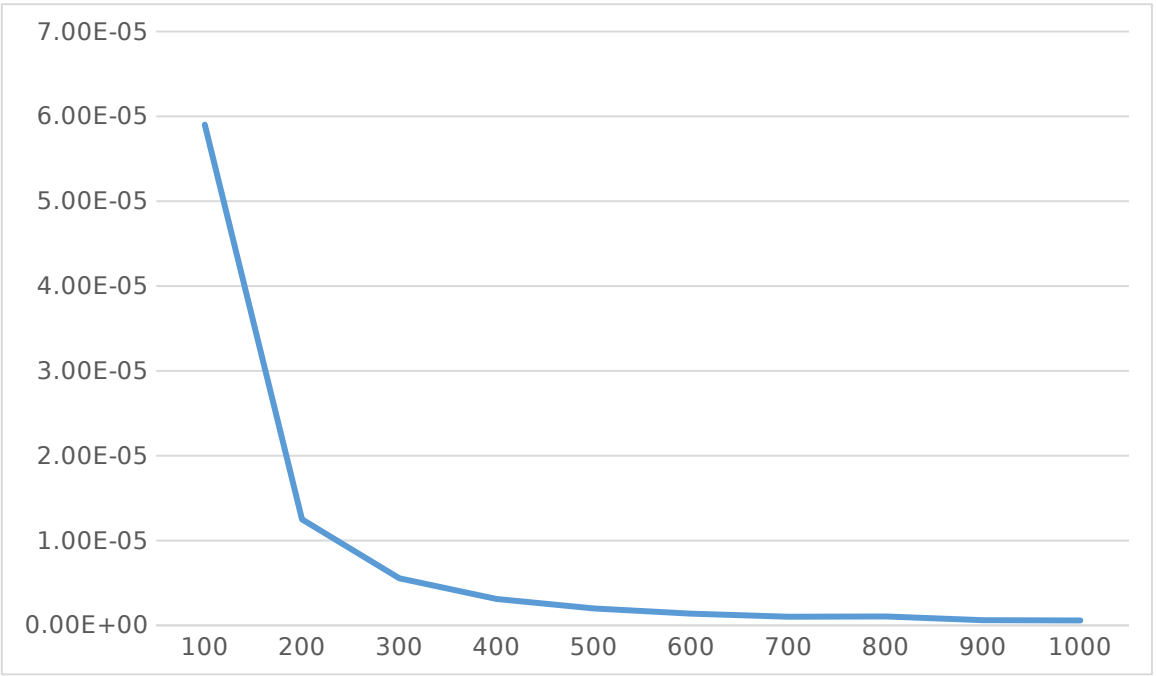
Рисунок 1. Пример выполнения программы 1

Зависимость погрешности решения от величины шагов сетки:

N	t	Время	Погрешность
100	5e-05	0.001	0,00433715
500	2e-06	0.013	0,00094166
1000	5e-07	0.045	0,00047947

Таблица 1. Зависимость погрешности и времени от шага пространственно-временной сетки

Если нарушается условие устойчивости $\tau \le h^2/2$, то погрешность будет стремиться к бесконечности



Задача 2 (4 балла).

1) Написать вычислительную программу на языке программирования C++ решения задачи (1)-(5) по полностью неявной схеме и схеме Кранка-Николсон на равномерной сетке.
2) Выполнить сравнение точности получаемого решения по двум схемам с использованием точного решения. Построить графики погрешностей как функций координат и времени, а также графики норм погрешностей как функций шагов сетки.
3) Сравнить время решения задач по трем схемам (явной, полностью неявной и Кранка-Николсон), обеспечивающих получение решения с одинаковым уровнем погрешности.

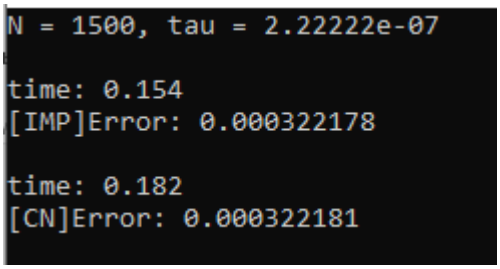
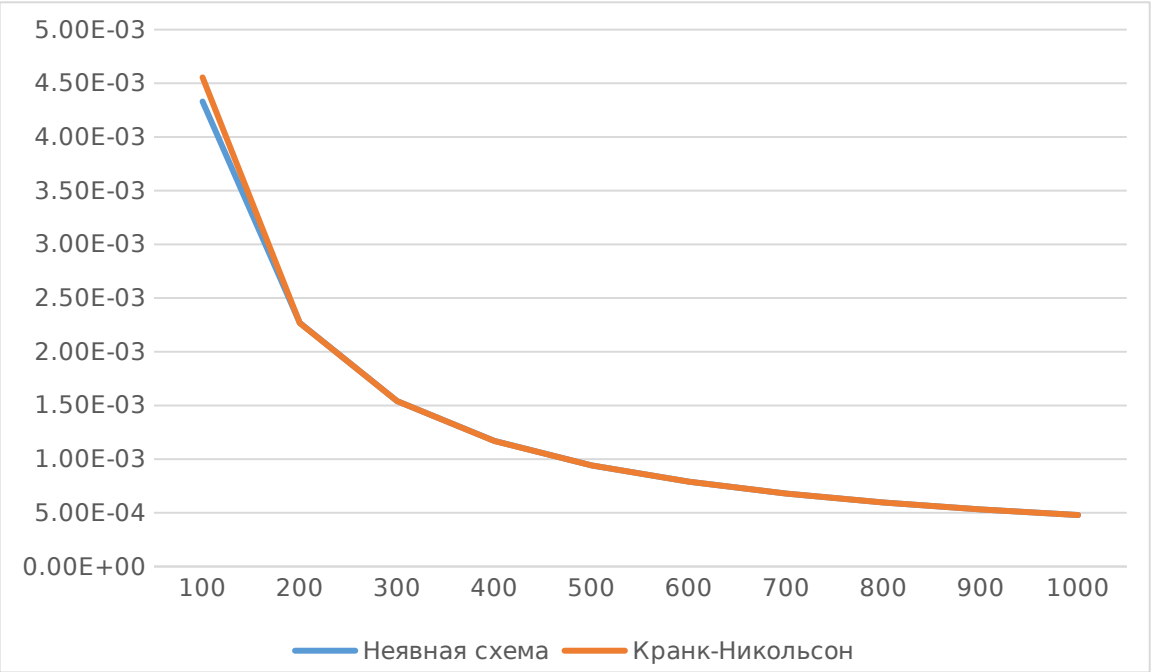


Рисунок 2. Пример выполнения программы 2

N	tau	Неявная схема	Время	Схема Кранка-Николсона	Время
100	5e-05	0.00433345	0.001	0.00433528	0.001
500	2e-06	0.00094159	0.015	0.00094163	0.02
1000	5e-07	0.00047946	0.068	0.000479467	0.081



По данным в таблице 2 можно сделать вывод, что время выполнения алгоритмов и погрешность примерно одинаковые.

Задача 3 (2 балла).

1) Написать вычислительную программу на языке программирования C++ решения задачи (1)-(4) с использованием консервативной схемы на равномерной сетке.
2) Убедиться в корректности программы на примере задачи 1.
3) Исследовать зависимость получаемого решения от величины шага сетки по пространственной и временной переменным. Построить графики решений для различных значений шага.

```
N = 100, tau = 1e-06

time: 0.002
Error: 0.00019545
```

Примем τ = 1E-10. Будем варьировать значение N, получим график, где на оси абсцисс указано значение N, а на оси ординат – значение погрешности при данном N.

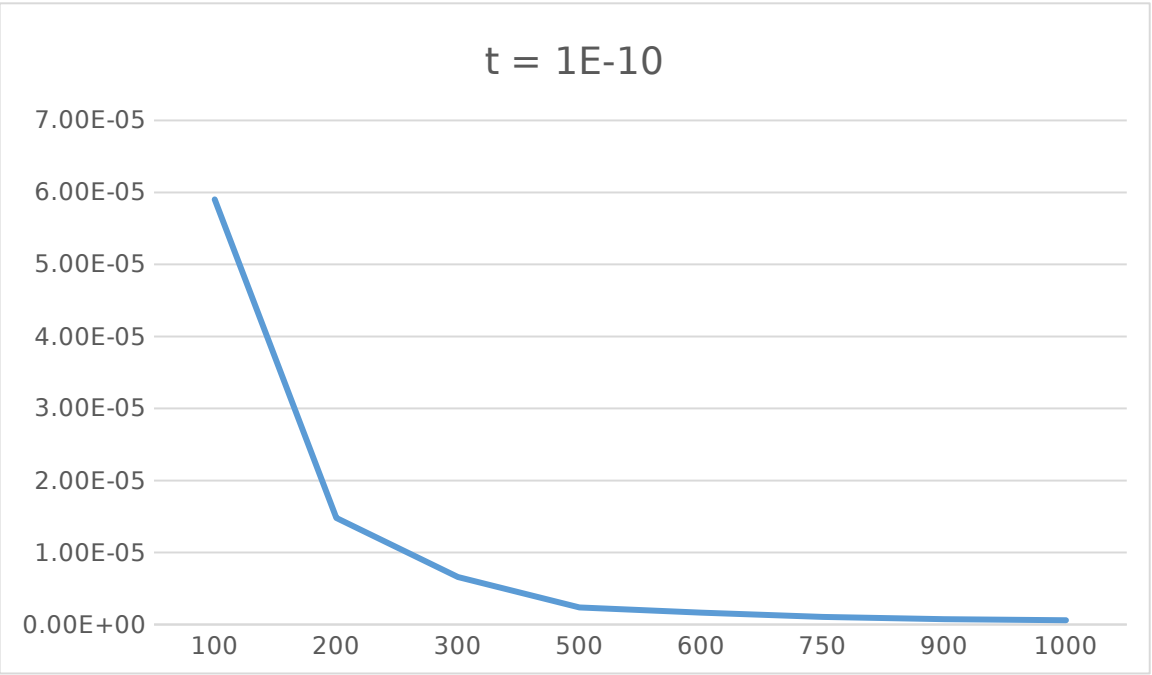


Рисунок 4. График зависимости погрешности решения от размера пространственной сетки

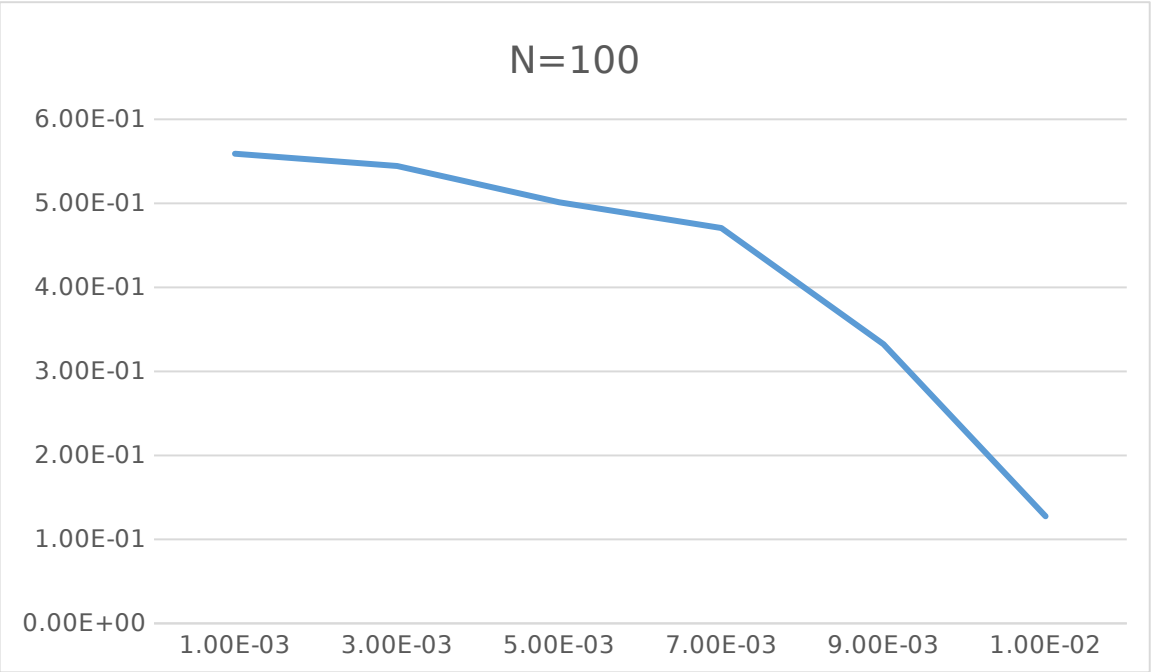


Рисунок 5. График зависимости погрешности решения от шага временной сетки

Исследуем также зависимость погрешности от шага сетки. Будем варьировать шаг сетки от 0.1e-3 до 0.01, при N=100. По полученным данным построим график, где на оси абсцисс указан шаг пространственно-временной сетки, а на оси ординат – значение погрешности. При увеличении шага погрешность уменьшается.

Вывод

В результате проделанной лабораторной работы был изучен теоретический материал необходимый для численного решения линейных и нелинейных начально-краевых задач для уравнений параболического типа с использованием различных конечно-разностных схем на примере задачи для одномерного уравнения теплопроводности с источником.

В ходе выполнения лабораторной работы был получен навык численного решения линейных и нелинейных начально-краевых задач для уравнений параболического типа с использованием различных конечно-разностных схем на примере задачи для одномерного уравнения теплопроводности с источником. Построены соответствующие графики решений и погрешностей.

Для каждой поставленной задачи написана вычислительная программа на языке программирования C++, выполняющая необходимые построения и расчеты.

Приложение

```
#include <iostream>
#include <cmath>
#include <iostream>
#include <fstream>
#include <ctime>

#define a0 1
#define a1 1
#define b0 -1
#define b1 0

using namespace std;

double F(double x, double t)
{
	return exp(-t);
}
double fi(double x)
{
	return ((sin(x) + cos(x)) / sin(1)) - x;
}
double psi_0(double t)
{
	return exp(-t);
}
double psi_1(double t)
{
	return exp(-t) * 1/tan(1);
}
double Solution(double x, double t)
{
	return exp(-t) * (((sin(x) + cos(x)) / sin(1)) - x);

}
double Error(double* y, double* y_sol, int N, double h, double tau)
{
	double maxdelta = -1.0;
	for (int j = 0; j <= N; j++)
	{
		for (int i = 0; i <= N; i++)
		{
			double delta = abs(y[j * (N + 1) + i] - y_sol[j * (N + 1) + i]);
			if (delta > maxdelta)
				maxdelta = delta;
		}
	}
	return maxdelta;
}
void Progonka(double* A, double* y, int j, int N)
{
	for (int i = 1; i < N; i++)
	{
		A[i + 1 * N] = A[i + 1 * N] - A[i - 1 + 0 * N] * (A[i - 1 + 2 * N] /
A[i - 1 + 1 * N]);
		A[i + 3 * N] = A[i + 3 * N] - A[i - 1 + 3 * N] * (A[i - 1 + 2 * N] /
A[i - 1 + 1 * N]);
		A[i - 1 + 2 * N] = 0;
	}

	y[N - 1 + N * j] = A[N - 1 + 3 * N] / A[N - 1 + 1 * N];

	for (int i = N - 2; i >= 0; i--)
		y[i + N * j] = (A[i + 3 * N] - A[i + 0 * N] * y[i + 1 + N * j]) / A[i +
1 * N];
```

```
}

//ЗАДАНИЕ 1
void KRS(int N, double h, double tau, double* y, double A)
{

	float start = clock();
	for (int i = 0; i <= N; i++)
		y[0 * (N + 1) + i] = fi(i * h);

	for (int n = 0; n < N; n++)
	{
		for (int i = 1; i < N; i++)

			y[(n + 1) * (N + 1) + i] = y[n * (N + 1) + i] + tau * F(i * h, n
* tau) +
			A * (tau / (h * h)) * (y[n * (N + 1) + i + 1] - 2.0 * y[n * (N +
1) + i] + y[n * (N + 1) + i - 1]);

		y[(n + 1) * (N + 1) + 0] = (h * psi_0((n + 1) * tau) - b0 * y[(n + 1) *
(N + 1) + 1]) / (a0 * h - b0);

		y[(n + 1) * (N + 1) + N] = (h * psi_1((n + 1) * tau) + b1 * y[(n + 1) *
(N + 1) + N - 1]) / (a1 * h + b1);
	}
	cout << "\ntime: " << (clock() - start) / CLOCKS_PER_SEC << endl;
}

// ЗАДАНИЕ 2
void KRS_IMP(int N, double h, double tau, double* y, double a)
{

	for (int i = 0; i <= N; i++)
		y[0 * (N + 1) + i] = fi(i * h);
	float start = clock();

	for (int j = 0; j < N; j++)
	{
		double* A = new double[(N + 1) * 4];
		A[0 * (N + 1) + 0] = b0;
		A[1 * (N + 1) + 0] = a0 * h - b0;
		A[1 * (N + 1) + N] = a1 * h + b1;
		A[2 * (N + 1) + N - 1] = -b1;
		A[3 * (N + 1) + 0] = h * psi_0(tau * (j + 1));
		A[3 * (N + 1) + N] = h * psi_1(tau * (j + 1));
		for (int i = 1; i < N; i++)
		{
			A[0 * (N + 1) + i] = a * tau / (h * h);
			A[1 * (N + 1) + i] = -1 - (a * 2 * tau / (h * h));
			A[2 * (N + 1) + i - 1] = a * tau / ((h * h));
			A[3 * (N + 1) + i] = -y[j * (N + 1) + i] - tau * F(i * h, tau *
j);
		}




		Progonka(A, y, j + 1, N + 1);
		delete[] A;
	}
	cout << "\ntime: " << (clock() - start) / CLOCKS_PER_SEC << endl;
}
void KRS_CN(int N, double h, double tau, double* y, double a)
{
```

```
	for (int i = 0; i <= N; i++)
		y[0 * (N + 1) + i] = fi(i * h);
	float start = clock();
	for (int j = 0; j < N; j++)
	{
		double* A = new double[(N + 1) * 4];
		A[0 * (N + 1) + 0] = b0;
		A[1 * (N + 1) + 0] = a0 * h - b0;
		A[1 * (N + 1) + N] = a1 * h + b1;
		A[2 * (N + 1) + N - 1] = -b1;
		A[3 * (N + 1) + 0] = h * psi_0(tau * (j + 1));
		A[3 * (N + 1) + N] = h * psi_1(tau * (j + 1));
		for (int i = 1; i < N; i++)
		{
			int k = j * (N + 1) + i;
			A[0 * (N + 1) + i] = a * tau / (2.0 * h * h);
			A[1 * (N + 1) + i] = -(1 + (a * tau / (h * h)));
			A[2 * (N + 1) + i - 1] = a * tau / (2.0 * (h * h));
			A[3 * (N + 1) + i] = -((1 - a * tau / (h * h)) * y[k] + a * tau /
(2.0 * (h * h)) * (y[k - 1] + y[k + 1]) + tau * F(i * h, tau * j));
		}


		Progonka(A, y, j + 1, N + 1);

		delete[] A;
	}
	cout << "\ntime: " << (clock() - start) / CLOCKS_PER_SEC << endl;
	return;
}

// ЗАДАНИЕ 3
double K(double u)
{
	return cos(u);
}
void KRS_CONS(int N, double h, double tau, double* y, double a)
{

	double ke, kw;
	double* x = new double[N + 1];
	double* t = new double[N + 1];

	for (int i = 0; i <= N; i++)
	{
		y[0 * (N + 1) + i] = fi(i * h);
		x[i] = i * h;
		t[i] = i * tau;

	}

	float start = clock();

	for (int j = 0; j < N; j++)
	{

		double* A = new double[(N + 1) * 4];

		A[0 * (N + 1) + 0] = b0;
		A[1 * (N + 1) + 0] = a0 * h - b0;
		A[1 * (N + 1) + N] = a1 * h + b1;
		A[2 * (N + 1) + N - 1] = -b1;
		A[3 * (N + 1) + 0] = h * psi_0(tau * (j + 1));
		A[3 * (N + 1) + N] = h * psi_1(tau * (j + 1));
```

```
		for (int i = 1; i < N; i++)
		{
			int k = j * (N + 1) + i;
			//ПРОВЕРКА
		   /* kw = 1;
			ke = 1;
			A[0 * (N + 1) + i] = -a * kw / (h * h);
			A[1 * (N + 1) + i] = (1. / tau + (ke + kw) * (a / (h * h)));
			A[2 * (N + 1) + i - 1] = -a / ((h * h)) * ke;
			A[3 * (N + 1) + i] = ((1. / tau - (1. - a) * (ke + kw) / (h * h))
* y[k]
				+ (1. - a) * (y[k + 1] * ke - y[k - 1] * kw) / (h * h)
				+ F(i * h, tau * j));*/

			kw = K((y[k - 1] + y[k]) / 2.);
			ke = K((y[k + 1] + y[k]) / 2.);

			A[0 * (N + 1) + i] = -a * kw / (h * h);
			A[1 * (N + 1) + i] = (1. / tau + (ke + kw) * (a / (h * h)));
			A[2 * (N + 1) + i - 1] = -a / ((h * h)) * ke;
			A[3 * (N + 1) + i] = ((1. / tau - (1. - a) * (ke + kw) / (h * h))
* y[k]
				+ (1. - a) * (y[k + 1] * ke - y[k - 1] * kw) / (h * h)
				+ y[k] * F(i * h, tau * j));
		}

		Progonka(A, y, j + 1, N + 1);
		delete[] A;


	}

	cout << "\ntime: " << (clock() - start) / CLOCKS_PER_SEC << endl;

}


int main()
{
	int N = 1000;
	double a = 0., b = 1.;
	double h = (double)(b - a) / N;
	//cout << h << endl;
	double tau = h * h / 2;
	tau = 1E-10;
	double* y = new double[(N + 1) * (N + 1)];
	double* y_sol = new double[(N + 1) * (N + 1)];
	for (int j = 0; j <= N; j++)
	{
		for (int i = 0; i <= N; i++)
		{
			y_sol[j * (N + 1) + i] = Solution(i * h, j * tau);
		}
	}
	cout << "N = " << N << ", tau = " << tau << endl;
	//KRS(N, h, tau, y, 1);
	//cout << "Error: " << Error(y, y_sol, N, h, tau) << endl;
	KRS_IMP(N, h, tau, y, 1);
	cout << "[IMP]Error: " << Error(y, y_sol, N, h, tau) << endl;
	KRS_CN(N, h, tau, y, 1);
	cout << "[CN]Error: " << Error(y, y_sol, N, h, tau) << endl;
	KRS_CONS(N, h, tau, y, 1);
	cout << "Error: " << Error(y, y_sol, N, h, tau) << endl;


	return 0;
}
```