САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ТЕХНИЧЕСКОЙ КИБЕРНЕТИКИ
КАФЕДРА "ИНФОРМАЦИОННЫХ И УПРАВЛЯЮЩИХ СИСТЕМ"

Отчет
по курсовой работе
по предмету “Математические модели технических объектов”

Выполнил:
Бурков С.К.
Гр. 3084/3
Проверил:
доц. С.П.Воскобойников.

2005

Содержание

Постановка задачи

Используя интегро - интерполяционный метод, разработать программу для моделирования распределения температуры в брусе, описываемого математической моделью

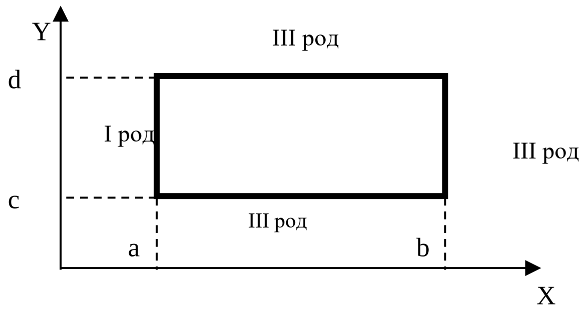
$$-\left[\frac{\partial}{\partial x}\left(k_1(x,y)\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(k_2(x,y)\frac{\partial u}{\partial y}\right)\right]=f(x,y),$$

$$a \le x \le \mathrm{b},\quad \mathrm{c} \le y \le \mathrm{d},\quad 0<\mathrm{c}_{11} \le k_1(\mathrm{x,y}) \le c_{12},\quad 0<\mathrm{c}_{21} \le k_2(\mathrm{x,y}) \le c_{22}$$

с граничными условиями:

$$u|_{x=a}=g_1(y),\qquad -k_1\frac{\partial u}{\partial x}|_{x=b}=\chi_2 u|_{x=b}-g_2(y),$$
$$k_2\frac{\partial u}{\partial y}|_{y=c}=\chi_3 u|_{y=c}-g_3(x),\qquad -k_2\frac{\partial u}{\partial y}|_{y=d}=\chi_4 u|_{y=d}-g_4(x),\qquad c_i^{3}0,\ \mathrm{i}=1,2,3,4.$$

Вид области и расположение граничных условий:



Для решения системы линейных алгебраических уравнений использовать метод сопряженных градиентов с предобусловливанием, причем матрица алгебраической системы должна храниться в упакованной форме.

Дискретная модель.

Построим дискретную модель для данной двумерной модели.

$a=x_L<x_1< \ldots <x_R=b$

$c=y_L<y_1< \ldots <y_R=d$

Введем в прямоугольнике $[x_L, x_R]\times[y_L, y_R]$ равномерную для простоты сетку, то есть множество узлов с координатами

$$x_i = x_L + ih_x,\quad i=0,1,2,\ldots,N_x$$
$$y_j = y_L + jh_y,\quad j=0,1,2,\ldots,N_y \qquad (2)$$

где $h_x = \frac{x_R - x_L}{N_x}$, $h_y = \frac{y_L - y_R}{N_y}$, N_x и N_y - числа разбиений по x и по y. Кроме этой основной сетки, введем вспомогательную сетку с координатами $x_{i-1/2} = \frac{x_i + x_{i-1}}{2}$, $y_{j-1/2} = \frac{y_j + y_{j-1}}{2}$.

$$\hbar_i = \left\{\frac{h}{2}, i = 0\right| \quad \left\{\frac{h+h}{2}, i = 1 \ldots N_x - 1\right| \qquad \hbar_j = \left\{\frac{h}{2}, j = 0\right| \quad \left\{\frac{h+h}{2}, j = 1 \ldots N_y - 1\right|$$

При решении ДУ будем использовать аппроксимацию решения и граничных условий второго порядка.

Интегрируем исходное ДУ второго порядка в промежутке $x \in [x_{i-1/2}, x_{i+1/2}]$, $y \in [y_{i-1/2}, y_{i+1/2}]$ и, аппроксимируя полученные производные, получаем систему алгебраических уравнений для всех внутренних точек области:

для $i=1,2,\ldots, N_x-1$; $j=1,2,\ldots, N_y-1$:

$$-[h_y k_1(x_{i+1/2}, y_j)\frac{u_{i+1,j}-u_{i,j}}{h_x} - h_y k_1(x_{i-1/2}, y_j)\frac{u_{i,j}-u_{i-1,j}}{h_x} +$$
$$+h_x k_2(x_i, y_{j+1/2})\frac{u_{i,j+1}-u_{i,j}}{h_y} - h_x k_2(x_i, y_{j-1/2})\frac{u_{i,j}-u_{i,j-1}}{h_y}] = h_x h_y f_{ij} \qquad (1)$$

Теперь рассмотрим вид этого уравнения для границ области. При этом соответствующие производные заменяются граничными условиями.

Для правой вертикальной границы (x=xR).

$$-[-h_y\left(c_2(y_j)u_{N_x,j} - \nu_2(y_j)\right) - h_y k_1(x_{N_x-1/2}, y_j)\frac{u_{N_x,j}-u_{N_x-1,j}}{h_x} + \frac{h_x}{2}k_2(x_{N_x}, y_{j+1/2})\frac{u_{N_x,j+1}-u_{N_x,j}}{h_y} -$$
$$-\frac{h_x}{2}k_2(x_{N_x}, y_{j-1/2})\frac{u_{N_x,j}-u_{N_x,j-1}}{h_y} - \frac{h_x}{2}h_y q_{N_x,j}u_{N_x,j}] = \frac{h_x}{2}h_y f_{N_x,j} \qquad (2)$$

$i=N_x$; $j=1,2,\ldots,N_y-1$;

Для аппроксимации краевых условий при $y=y_L$ и $y=y_R$ уравнение (1) интегрируется по ячейкам $[x_{i-1/2}, x_{i+1/2}]´[y_0, y_{1/2}]$ и $[x_{i-1/2}, x_{i+1/2}]´[y_{N-1/2}, y_N]$ соответственно. С учетом граничных условий это дает следующие соотношения

$$-[\frac{h_y}{2}k_1(x_{i+1/2}, y_0)\frac{u_{i+1,0}-u_{i,0}}{h_x} - \frac{h_y}{2}k_1(x_{i-1/2}, y_0)\frac{u_{i,0}-u_{i-1,0}}{h_x} +$$
$$+h_x k_2(x_i, y_{1/2})\frac{u_{i,1}-u_{i,0}}{h_y} - h_x\left(c_3(x_i)u_{i,0} - \nu_3(x_i)\right) - h_x\frac{h_y}{2}q_{i,0}u_{i,0}] = h_x\frac{h_y}{2}f_{i,0} \qquad (3)$$

$i=1,2,\ldots, N_x-1$; $j=0$.

$$-[\frac{h_y}{2}k_1\left(x_{i+1/2}, y_{N_y}\right)\frac{u_{i+1,N_y}-u_{i,N_y}}{h_x} - \frac{h_y}{2}k_1\left(x_{i-1/2}, y_{N_y}\right)\frac{u_{i,N_y}-u_{i-1,N_y}}{h_x} - h_x\left(c_4(x_i)u_{i,N_y} - \nu_4(x_i)\right) -$$
$$-h_x k_2\left(x_i, y_{N_y-1/2}\right)\frac{u_{i,N_y}-u_{i,N_y-1}}{h_y} - h_x\frac{h_y}{2}q_{i,N_y}] = h_x\frac{h_y}{2}f_{i,N_y} \qquad (4)$$

$i=1,2,\ldots,N_y-1$; $j = N_y$.

Теперь напишем аппроксимацию граничных условий в угловых точках прямоугольника. В этом случае на граничные условия заменяются две производные.

$$-[-\frac{h_y}{2}\left(c_2(y_0)u_{N_x,0} - \nu_2(y_0)\right) - \frac{h_y}{2}k_1(x_{N_x-1/2}, y_0)\frac{u_{N_x,0}-u_{N_x-1,0}}{h_x} + \frac{h_x}{2}k_2(x_{N_x}, y_{1/2})\frac{u_{N_x,1}-u_{N_x,0}}{h_y}$$
$$-\frac{h_x}{2}\left(c_3(x_{N_x})u_{N_x,0} - \nu_3(x_{N_x})\right) - \frac{h_x}{2}\frac{h_y}{2}q_{N_x,0}u_{N_x,0}] = \frac{h_x}{2}\frac{h_y}{2}f_{N_x,0} \qquad (5)$$

$i=N_x$; $j=0$

$$-[-\frac{h_y}{2}\left(c_2\left(y_{N_y}\right)u_{N_x,N_y}-\nu_2\left(y_{N_y}\right)\right)-\frac{h_y}{2}k_1\left(x_{N_y-1/2},y_{N_y}\right)\frac{u_{N_x,N_y}-u_{N_x-1,N_y}}{h_x}-\frac{h_x}{2}(c_4(x_{N_x})u_{N_x,N_y}-$$
$$-\nu_4(x_{N_x}))-\frac{h_x}{2}k_2\left(x_{N_x},y_{N_y-1/2}\right)\frac{u_{N_x,N_y}-u_{N_x,N_y-1}}{h_y}-\frac{h_x}{2}\frac{h_y}{2}q_{N_x,N_y}u_{N_x,N_y}]=\frac{h_x}{2}\frac{h_y}{2}f_{N_x,N_y} \qquad (6)$$

i=N $_x$; j=N $_y$

Теперь рассмотрим первое разбиение по X (i=0). Здесь (i+1)-я компонента в алгебраическом уравнении заменяется на граничное условие первого рода – известную функцию V1(y). соответственно.

Для решения системы (1)-(6) перенумеруем узлы разбиения по оси X. Для этого сделаем следующую замену:

$V_{i,j} = W_k$
$V_{i-1,j} = W_{k-1}$
$V_{i+1,j} = W_{k+1}$
$V_{i,j-1} = W_{k-L}$
$V_{i,j+1} = W_{k+L}$, где $L = N_x$
При такой замене нумерация узлов области будет определяться следующей формулой:

$k = j*N_x + i - 1$,
где $i = 1..N_x$; $j = 0..N_y$.
При этом размерность системы будет задаваться так:
$N=N_x*(N_y+1)$.

Решение системы неявным методом сопряженных градиентов.

Получаемую в результате замены систему (1) - (6) будем решать неявным методом сопряженных градиентов, который решает систему алгебраических уравнений

A*X=F
следующим образом:

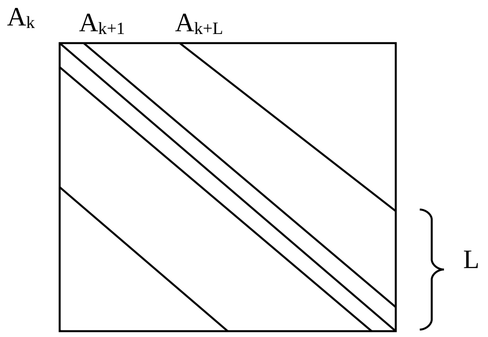
$$r_0 = F - \mathrm{Ax}_0$$
$$\mathrm{BW}_0 = r_0$$
$$S_1 = W_0,$$
$$k = 1...N$$
$$[\alpha_k = \frac{(W_{k-1}, r_{k-1})}{(\mathrm{AS}_k, S_k)}$$
$$[x_k = x_{k-1} + \alpha_k S_k$$
$$[r_k = r_{k-1} - \alpha_k \mathrm{AS}_k$$
$$[\frac{\|r_k\|}{\|F\|} < \varepsilon \qquad [$$
$$[\mathrm{BW}_k = r_k$$
$$[\beta_k = \frac{(W_k, r_k)}{(W_{k-1}, r_{k-1})}$$
$$[S_{k+1} = W_k + \beta_k S_k$$

При этом условием окончания итерационного процесса является удовлетворение неравенства:

$\frac{\|r_k\|}{\|b\|} < \varepsilon$, где ε - задаваемая точность решения.

Матрица A будет квадратная, пятидиагональная (остальные элементы нулевые), симметричная и имеет следующую структуру:

A =



Вследствие такого вида матрицы A будем хранить только три диагонали: A_k, A_{k+1} и A_{k+L}.

При этом в методе сопряженных градиентов:

$$B = LL^T$$

$$\{Ly_k = r_k|$$

Размерность матрицы A будет определяться количеством узлов, т.е.

$N=N_x*(N_y+1)$

Соответствующие элементы диагоналей Ak, Bk и Ck и b для каждого узла k (i и j):

i=1..N_x-1, j=1..N_y-1

$$\mathrm{Ak}[k] = \frac{\mathrm{hy}}{\mathrm{hx}} \cdot k1(x_{i+1/2}, y_j) + \frac{\mathrm{hy}}{\mathrm{hx}} \cdot k1(x_{i-1/2}, y_j) + \frac{\mathrm{hx}}{\mathrm{hy}} \cdot k2(x_i, y_{j+1/2}) + \frac{\mathrm{hx}}{\mathrm{hy}} \cdot k2(x_i, y_{j-1/2})$$

$$\mathrm{Bk}[k] = -\frac{\mathrm{hy}}{\mathrm{hx}} \cdot k1(x_{i+1/2}, y_j)$$

$$\mathrm{Ck}[k] = -\frac{\mathrm{hx}}{\mathrm{hy}} \cdot k2(x_i, y_{j+1/2})$$

$$\mathrm{Fk}[k] = \mathrm{hx} \cdot \mathrm{hy} \cdot \mathrm{fun}(x_i, y_j)$$

$$\mathrm{if}(i = 1) \Rightarrow \mathrm{Fk}[k] = \mathrm{Fk}[k] + \frac{\mathrm{hy}}{\mathrm{hx}} \cdot V1(y_j) \cdot k1(x_{i-1/2}, y_j)$$

i=1, j=1,..Ny-1

$$\mathrm{Ak}[k] = \frac{\mathrm{hy}}{\mathrm{hx}} \cdot k1(x_{i-1/2}, y_j) + \mathrm{hy} \cdot \mathrm{kappa2} + \frac{\mathrm{hx}}{2 \cdot \mathrm{hy}} \cdot (k2(x_i, y_{j+1/2}) + k2(x_i, y_{j-1/2}))$$

$$\mathrm{Ck}[k] = -\frac{\mathrm{hx}}{2 \cdot \mathrm{hy}} \cdot k2(x_i, y_{j+1/2})$$

$$\mathrm{Fk}[k] = \frac{\mathrm{hx}}{2} \cdot \mathrm{hy} \cdot \mathrm{fun}(x_i, y_j) + \mathrm{hy} \cdot V2(y_j)$$

j=Ny, i=1,..Nx-1

$$\mathrm{Ak}[k] = \frac{\mathrm{hy}}{2 \cdot \mathrm{hx}} \cdot (k1(x_{i+1/2}, y_j) + k1(x_{i-1/2}, y_j)) + \mathrm{hx} \cdot (\frac{k2(x_i, y_{j-1/2})}{\mathrm{hy}} + \mathrm{kappa4})$$

$$\mathrm{Bk}[k] = -\frac{\mathrm{hy}}{2 \cdot \mathrm{hx}} \cdot k1(x_{i+1/2}, y_j)$$

$$\mathrm{Fk}[k] = \mathrm{hx} \cdot \frac{\mathrm{hy}}{2} \cdot \mathrm{fun}(x_i, y_j) + V4(x_i) \cdot \mathrm{hx}$$

$$\mathrm{if}(i = 1) \Rightarrow \mathrm{Fk}[k] = \mathrm{Fk}[k] + \frac{\mathrm{hy}}{2 \cdot \mathrm{hx}} \cdot V1(y_j) \cdot k1(x_{i-1/2}, y_j)$$

j=0, i=1,..Nx-1

$$\mathrm{Ak}[k] = \frac{\mathrm{hy}}{2\cdot\mathrm{hx}}\cdot(k1(x_{i+1/2},y_j) + k1(x_{i-1/2},y_j)) + \mathrm{hx}\cdot(\frac{k2(x_i,y_{j+1/2})}{\mathrm{hy}} + \mathrm{kappa3})$$

$$\mathrm{Bk}[k] = -\frac{\mathrm{hy}}{2\cdot\mathrm{hx}}\cdot k1(x_{i+1/2},y_j)$$

$$\mathrm{Ck}[k] = -\frac{\mathrm{hx}}{2\cdot\mathrm{hy}}\cdot k2(x_i,y_{j+1/2})$$

$$\mathrm{Fk}[k] = \mathrm{hx}\cdot\frac{\mathrm{hy}}{2}\cdot\mathrm{fun}(x_i,y_j) + V3(x_i)\cdot\mathrm{hx}$$

$$\mathrm{if}(i=1) \Rightarrow \mathrm{Fk}[k] = \mathrm{Fk}[k] + \frac{\mathrm{hy}}{2\cdot\mathrm{hx}}\cdot V1(y_j)\cdot k1(x_{i-1/2},y_j)$$

i=Nx, j=0

$$\mathrm{Ak}[k] = \frac{\mathrm{hy}}{2\cdot\mathrm{hx}}\cdot k1(x_{i-1/2},y_j) + \frac{\mathrm{hy}}{2}\cdot\mathrm{kappa2} + \frac{\mathrm{hx}}{2\cdot\mathrm{hy}}\cdot k2(x_i,y_{j+1/2}) + \frac{\mathrm{hx}}{2}\cdot\mathrm{kappa3}$$

$$\mathrm{Ck}[k] = -\frac{\mathrm{hx}}{2\cdot\mathrm{hy}}\cdot k2(x_i,y_{j+1/2})$$

$$\mathrm{Fk}[k] = \frac{\mathrm{hx}}{2}\cdot\frac{\mathrm{hy}}{2}\cdot\mathrm{fun}(x_i,y_j) + \frac{\mathrm{hy}}{2}\cdot V2(y_j) + \frac{\mathrm{hx}}{2}\cdot V3(x_i)$$

i=Nx, j=Ny

$$\mathrm{Ak}[k] = \frac{\mathrm{hy}}{2\cdot\mathrm{hx}}\cdot k1(x_{i-1/2},y_j) + \frac{\mathrm{hy}}{2}\cdot\mathrm{kappa2} + \frac{\mathrm{hx}}{2\cdot\mathrm{hy}}\cdot k2(x_i,y_{j-1/2}) + \frac{\mathrm{hx}}{2}\cdot\mathrm{kappa4}$$

$$\mathrm{Fk}[k] = \frac{\mathrm{hx}}{2}\cdot\frac{\mathrm{hy}}{2}\cdot\mathrm{fun}(x_i,y_j) + \frac{\mathrm{hy}}{2}\cdot V2(y_j) + \frac{\mathrm{hx}}{2}\cdot V4(x_i)$$

Тесты для заданной модели

Во всех тестах вектор начальных приближений возьмем нулевой.

Тест №1

Первый тест определяет решение интегро - интерполяционного метода без погрешности и имеет вид констант:

U(x,y) = 3;
f(x,y) = 0;
$g_1(y) = 3$;
$g_2(y) = 3$;
$g_3(x) = 3$;
$g_4(x) = 3$;
$k_1(x,y) = 2$; $k_2(x,y) = 5$
$\chi_1 = 1$; $\chi_3 = 1$;
$x \in [0,1]$ $y \in [0,1]$

Для анализа погрешности аппроксимации при увеличении числа разбиений проведем несколько тестов.

Полученные результаты представлены в таблице:

k – количество итераций	Nx	Ny	N- порядок матрицы	Погрешность
11	4	4	20	2,78935097242083E-10
16	8	8	72	6,54250920106847E-09

28	16	16	272	9,35265154211606E-09
50	32	32	1056	1,81024795153917E-08
96	64	64	4160	2,91886177450351E-08

По результатам таблицы видно, что с увеличением количества шагов погрешность (вычислительная) возрастает из-за накапливающихся ошибок округления

Тест №2

Для второго теста без погрешности аппроксимации были использованы следующие функции:

U(x,y) = `2*x+3*y+5;`
f(x,y) = `-10;`
g_1(y) = `2*xLeft+3*y+5;`
g_2(y) = `kappa2*(2*xRight+3*y+5)+(2*xRight+y+1)*2;`
g_3(x) = `kappa3*(2*x+3*yLeft+5)-(x+2*yLeft+1)*3;`
g_4(x) = `kappa4*(2*x+3*yRight+5)+(x+2*yRight+1)*3;`
k_1(x,y) = `2*x + y + 1;`
k_2(x,y) = `x + 2*y + 1;`
$\chi_1 = 1$; $\chi_3 = 1$;
$x \in [0,1]$ $y \in [0,1]$

Для анализа погрешности аппроксимации при увеличении числа разбиений проведем несколько тестов. Полученные результаты:

k – количество итераций	Nx	Ny	N- порядок матрицы	Погрешность
10	4	4	20	1,15544658285671E-08
16	8	8	72	1,0345834411396E-08
28	16	16	272	1,17305702929116E-08
50	32	32	1056	4,26411101983604E-08
99	64	64	4160	1,15548547618971E-07

В данном примере хорошо видно, что функция выходит на свое стационарное значение.

По результатам таблицы видно, что с увеличением количества шагов погрешность (вычислительная) возрастает из-за накапливающихся ошибок округления.

Тест №3

Для получения погрешности аппроксимации воспользуемся следующими функциями:

Зададим:
U(x,y) = `2*x*x + 3*y*y*y + 4;`
k_1(x,y) = `1 + 2*x*x + y;`
k_2(x,y) = `1 + x + 2*y*y;`
$\chi_1 = 1$; $\chi_3 = 1$;
$x \in [0,2]$ $y \in [0,2]$

Получаем следующие функции:
f(x,y) = `-(24*x*x + 18*x*y + 22*y + 72*y*y*y + 4);`
g_1(y) = `2*xLeft*xLeft+3*y*y*y + 4;`
g_2(y) = `kappa2*(2*xRight*xRight+3*y*y*y+4)+(1+2*xRight*xRight+y)*4*xRight;`
g_3(x) = `kappa3*(2*x*x+4+3*yLeft*yLeft*yLeft)-(1+x+2*yLeft*yLeft)*9*yLeft*yLeft;`
g_4(x)=`kappa4*(2*x*x+4+3*yRight*yRight*yRight)+(1+x+2*yRight*yRight)*9*yRight*yRight;`

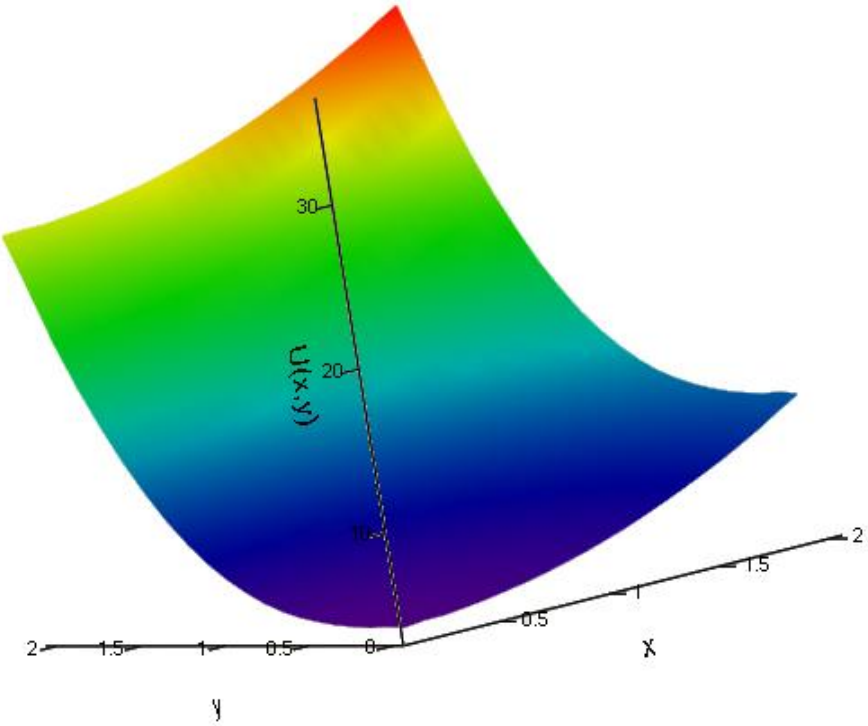
Nx = 16, Ny = 16; N = 272;

Полученные результаты:

x	y	U(x,y)	x	y	U(x,y)	x	y	U(x,y)	x	y	U(x,y)
0,125	0	4,02563738	0,625	0	4,732965	1,125	0	6,4530443	1,625	0	9,1882854
0,125	0,125	4,026978835	0,625	0,125	4,7294427	1,125	0,125	6,4482514	1,625	0,125	9,1835441
0,125	0,25	4,065006722	0,625	0,25	4,7611969	1,125	0,25	6,478061	1,625	0,25	9,2132173
0,125	0,375	4,173887974	0,625	0,375	4,8631199	1,125	0,375	6,5775169	1,625	0,375	9,3123995
0,125	0,5	4,388344757	0,625	0,5	5,0701259	1,125	0,5	6,7816978	1,625	0,5	9,5162215
0,125	0,625	4,743324091	0,625	0,625	5,4171829	1,125	0,625	7,1257223	1,625	0,625	9,8598516
0,125	0,75	5,273866685	0,625	0,75	5,9393139	1,125	0,75	7,6447447	1,625	0,75	10,378489
0,125	0,875	6,015049338	0,625	0,875	6,6715842	1,125	0,875	8,3739465	1,625	0,875	11,107354
0,125	1	7,001955531	0,625	1	7,6490849	1,125	1	9,3485279	1,625	1	12,081678
0,125	1,125	8,269656251	0,625	1,125	8,9069225	1,125	1,125	10,6037	1,625	1,125	13,336699
0,125	1,25	9,853192083	0,625	1,25	10,480213	1,125	1,25	12,17468	1,625	1,25	14,907654
0,125	1,375	11,78754962	0,625	1,375	12,404083	1,125	1,375	14,096689	1,625	1,375	16,829775
0,125	1,5	14,10762335	0,625	1,5	14,71367	1,125	1,5	16,404951	1,625	1,5	19,138291
0,125	1,625	16,84814788	0,625	1,625	17,444132	1,125	1,625	19,134689	1,625	1,625	21,868422
0,125	1,75	20,04357179	0,625	1,75	20,630655	1,125	1,75	22,321127	1,625	1,75	25,055384
0,125	1,875	23,72781605	0,625	1,875	24,308458	1,125	1,875	25,999486	1,625	1,875	28,734387
0,125	2	27,93380274	0,625	2	28,512798	1,125	2	30,204988	1,625	2	32,940636
0,25	0	4,109495742	0,75	0	5,0674601	1,25	0	7,0419762	1,75	0	10,02987
0,25	0,125	4,108889394	0,75	0,125	5,0634125	1,25	0,125	7,0371219	1,75	0,125	10,025229
0,25	0,25	4,144660806	0,75	0,25	5,0944064	1,25	0,25	7,0667906	1,75	0,25	10,054997
0,25	0,375	4,251283865	0,75	0,375	5,1953859	1,25	0,375	7,1660462	1,75	0,375	10,15427
0,25	0,5	4,463445969	0,75	0,5	5,4013208	1,25	0,5	7,3699886	1,75	0,5	10,358182
0,25	0,625	4,816014762	0,75	0,625	5,7472273	1,25	0,625	7,7137562	1,75	0,625	10,701904
0,25	0,75	5,343965371	0,75	0,75	6,2681684	1,25	0,75	8,2325214	1,75	0,75	11,220638
0,25	0,875	6,082324063	0,75	0,875	6,9992448	1,25	0,875	8,9614816	1,75	0,875	11,949606
0,25	1	7,066131164	0,75	1	7,9755825	1,25	1	9,9358503	1,75	1	12,924042
0,25	1,125	8,33041599	0,75	1,125	9,2323243	1,25	1,125	11,19085	1,75	1,125	14,179184
0,25	1,25	9,910177165	0,75	1,25	10,804626	1,25	1,25	12,761707	1,75	1,25	15,75027
0,25	1,375	11,84036383	0,75	1,375	12,727653	1,25	1,375	14,683649	1,75	1,375	17,672532
0,25	1,5	14,15585541	0,75	1,5	15,036586	1,25	1,5	16,991902	1,75	1,5	19,981196
0,25	1,625	16,89144084	0,75	1,625	17,766618	1,25	1,625	19,721692	1,75	1,625	22,711483
0,25	1,75	20,081805	0,75	1,75	20,952964	1,25	1,75	22,90824	1,75	1,75	25,898603
0,25	1,875	23,76154695	0,75	1,875	24,630853	1,25	1,875	26,586765	1,75	1,875	29,577763
0,25	2	27,96529463	0,75	2	28,835533	1,25	2	30,792482	1,75	2	33,784165
0,375	0	4,254513413	0,875	0	5,4656737	1,375	0	7,6942377	1,875	0	10,934381
0,375	0,125	4,252650176	0,875	0,125	5,4612589	1,375	0,125	7,6893807	1,875	0,125	10,92986
0,375	0,25	4,286723738	0,875	0,25	5,4916998	1,375	0,25	7,7189915	1,875	0,25	10,959754
0,375	0,375	4,391434994	0,875	0,375	5,5919812	1,375	0,375	7,8181491	1,875	0,375	11,059155
0,375	0,5	4,601547993	0,875	0,5	5,797118	1,375	0,5	8,0219682	1,875	0,5	11,263199
0,375	0,625	4,951937857	0,875	0,625	6,1421679	1,375	0,625	8,3656018	1,875	0,625	11,607056
0,375	0,75	5,477568188	0,875	0,75	6,6622308	1,375	0,75	8,884235	1,875	0,75	12,12593
0,375	0,875	6,213453974	0,875	0,875	7,3924396	1,375	0,875	9,6130763	1,875	0,875	12,855044
0,375	1	7,194630756	0,875	1	8,3679511	1,375	1	10,587349	1,875	1	13,829631
0,375	1,125	8,456133334	0,875	1,125	9,6239377	1,375	1,125	11,842282	1,875	1,125	15,084931
0,375	1,25	10,03298215	0,875	1,25	11,195583	1,375	1,25	13,413108	1,875	1,25	16,656179
0,375	1,375	11,96017565	0,875	1,375	13,118082	1,375	1,375	15,335057	1,875	1,375	18,578608
0,375	1,5	14,27268873	0,875	1,5	15,426635	1,375	1,5	17,643359	1,875	1,5	20,887442
0,375	1,625	17,00548008	0,875	1,625	18,156457	1,375	1,625	20,373238	1,875	1,625	23,6179
0,375	1,75	20,1935138	0,875	1,75	21,342768	1,375	1,75	23,559913	1,875	1,75	26,805191
0,375	1,875	23,87180353	0,875	1,875	25,020798	1,375	1,875	27,2386	1,875	1,875	30,484514
0,375	2	28,07548544	0,875	2	29,225785	1,375	2	31,44451	1,875	2	34,691063
0,5	0	4,462093731	1	0	5,9275687	1,5	0	8,4097097	2	0	11,901744
0,5	0,125	4,459287048	1	0,125	5,9229146	1,5	0,125	8,4048941	2	0,125	11,897364
0,5	0,25	4,492050464	1	0,25	5,952972	1,5	0,25	8,4345113	2	0,25	11,92741
0,5	0,375	4,595204604	1	0,375	6,0527585	1,5	0,375	8,5336503	2	0,375	12,026969
0,5	0,5	4,803595167	1	0,5	6,2573249	1,5	0,5	8,7374361	2	0,5	12,231175
0,5	0,625	5,152139467	1	0,625	6,6017632	1,5	0,625	9,0810311	2	0,625	12,5752
0,5	0,75	5,675823542	1	0,75	7,1212035	1,5	0,75	9,599629	2	0,75	13,094248
0,5	0,875	6,409680205	1	0,875	7,8508063	1,5	0,875	10,328446	2	0,875	13,823539
0,5	1	7,388767052	1	1	8,8257527	1,5	1	11,30271	2	1	14,798308
0,5	1,125	8,648151023	1	1,125	10,081238	1,5	1,125	12,557657	2	1,125	16,053792
0,5	1,25	10,22290021	1	1,25	11,652465	1,5	1,25	14,128521	2	1,25	17,625227

0,5	1,375	12,14808216	1	1,375	13,574646	1,5	1,375	16,050535	2	1,375	19,547844
0,5	1,5	14,45876828	1	1,5	15,882996	1,5	1,5	18,358928	2	1,5	21,856866
0,5	1,625	17,19004441	1	1,625	18,612735	1,5	1,625	21,088922	2	1,625	24,587511
0,5	1,75	20,37702771	1	1,75	21,799088	1,5	1,75	24,275735	2	1,75	27,774985
0,5	1,875	24,05488845	1	1,875	25,477281	1,5	1,875	27,954581	2	1,875	31,454485
0,5	2	28,2588717	1	2	29,68254	1,5	2	32,160668	2	2	35,661191

Полученный график функции представлен ниже:



Проанализируем сходимость решения, рассмотрев конкретную точку – x=1, y=1.

Число разбиений - Nx	Значение функции – U(x,y)	Разница между значениями
8	8.304526	-
16	8.825756	-0.52123
32	8.956114	-0.130358
64	8.989102	-0.032988

По приведенным результатам четко видна сходимость решения к стационарному решению, причем погрешность при увеличении числа разбиений в 2 раза уменьшается в 4 раза.

Для анализа погрешности аппроксимации при увеличении числа разбиений проведем несколько тестов.

Полученные результаты представлены в следующей таблице:

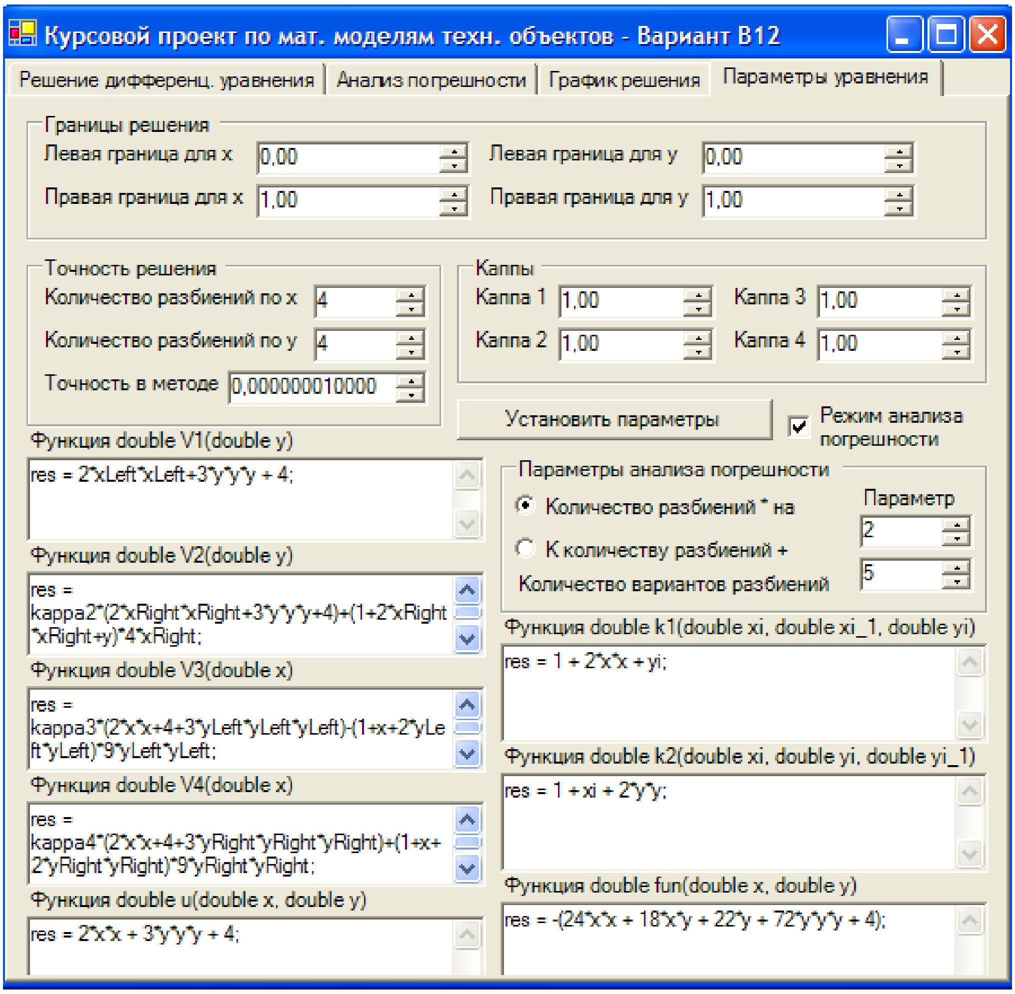
k – количество итераций	Nx	Ny	N- порядок матрицы	Погрешность
11	4	4	20	5,355776
17	8	8	72	1,359759
29	16	16	272	0,340835
54	32	32	1056	0,085266
106	64	64	4160	0,021321

Вывод

По результатам таблицы и графика можно видеть, что сначала погрешность достаточно велика (здесь оказывает влияние большая погрешность аппроксимации), затем погрешность аппроксимации постепенно затухает. Причем погрешность уменьшается как квадрат отношения числа разбиений, т.е. при увеличении числа разбиений в 2 раза, погрешность падает в 4.

Приложение 1

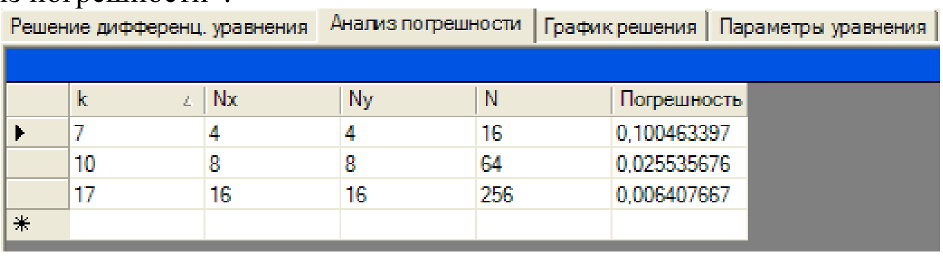
Интерфейс программы



Входными данными для программы являются:

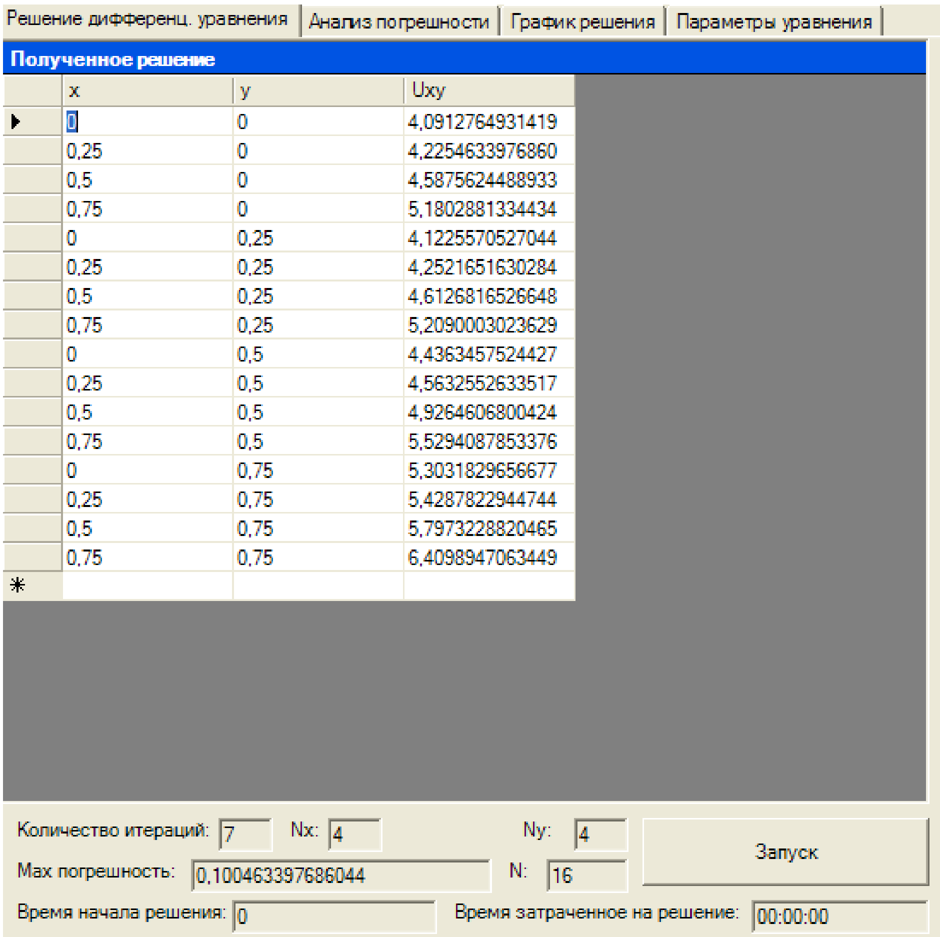
- Границы исследуемых диапазонов;
- Количество разбиений;
- Требуемая точность метода;
- Входные функции V1, V2, V3, V4, U, k1, k2, fun;

Также дополнительно можно установить режим анализа погрешности, позволяющий проследить изменение погрешности. Результаты по этому анализу заносятся в таблицу на вкладке “Анализ погрешности”:



k	Nx	Ny	N	Погрешность
7	4	4	16	0,100463397
10	8	8	64	0,025535676
17	16	16	256	0,006407667

Для получения результатов необходимо нажать кнопку “Запуск” на вкладке “Решение дифференц. уравнения”. Там же появляются результаты в таблице:



x	y	Uxy
0	0	4,0912764931419
0,25	0	4,2254633976860
0,5	0	4,5875624488933
0,75	0	5,1802881334434
0	0,25	4,1225570527044
0,25	0,25	4,2521651630284
0,5	0,25	4,6126816526648
0,75	0,25	5,2090003023629
0	0,5	4,4363457524427
0,25	0,5	4,5632552633517
0,5	0,5	4,9264606800424
0,75	0,5	5,5294087853376
0	0,75	5,3031829656677
0,25	0,75	5,4287822944744
0,5	0,75	5,7973228820465
0,75	0,75	6,4098947063449

На этой вкладке также выводится дополнительная информация, такая как

- максимальная погрешность
- количество итераций метода
- число разбиений Nx, Ny и N
- потраченное время

Текст программы

<u>Определение типов данных и методов</u>

```
using System;
using System.IO;
namespace MMKursProject
{
        public class Matrix
        {
                public int N;
                public int Nx;
                public bool fiveDiag;
                public bool econom;
                public bool transpon;
                public double[] A, B, C;
                public double[][] M;

                public Matrix(int N, int Nx, bool econom, bool fiveDiag)
                {
                        this.Nx = Nx;
                        this.N=N;
                        this.fiveDiag=fiveDiag;
                        this.econom=econom;
                        transpon=false;
                        if(econom)
                        {       A = new double[N];
                                B = new double[N - 1];
                                C = new double[N - Nx];
                        }
                        else
```

```
            {       M=new double[N][];
                    for(int i=0;i<N;i++)
                            M[i]=new double[N];
            }
    }
    public Matrix(int N, int Nx, string str)
    {
            this.Nx = Nx;
            this.N=N;
            transpon=false;
            if(str=="econom") this.econom=true;
            else if(str=="full") this.econom=false;
            else throw new ArgumentOutOfRangeException();
            if(econom)
            {
                    A = new double[N];
                    B = new double[N - 1];
                    C = new double[N - Nx];
            }
            else
            {
                    M=new double[N][];
                    for(int i=0;i<N;i++)
                            M[i]=new double[N];
            }
    }
    public Matrix(int N, int Nx, double[] A, double[] B, double[] C)
    {
            if ((A.Length!=N)||(B.Length!=N-1)||(C.Length!=N-Nx))
                    throw new ArgumentOutOfRangeException();
            this.Nx = Nx;
            transpon=false;
            this.N=N;
            this.econom=true;
            this.A = new double[N];
            this.B = new double[N - 1];
            this.C = new double[N - Nx];
            for (int i=0;i<N;i++)         this.A[i]=A[i];
            for (int i=0;i<N-1;i++)       this.B[i]=B[i];
            for (int i=0;i<N-Nx;i++)      this.C[i]=C[i];
    }
    public Matrix(int N, int Nx, Vector A, Vector B, Vector C)
    {
            if ((A.GetLenght!=N)||(B.GetLenght!=N-1)||(C.GetLenght!=N-Nx))
                    throw new ArgumentOutOfRangeException();
            this.Nx = Nx;
            this.N=N;
            transpon=false;
            this.econom=true;
            this.A = new double[N];
            this.B = new double[N - 1];
            this.C = new double[N - Nx];
            for (int i=0;i<N;i++)         this.A[i]=A[i];
            for (int i=0;i<N-1;i++)       this.B[i]=B[i];
            for (int i=0;i<N-Nx;i++)      this.C[i]=C[i];
    }
    public int GetLength
    {
            get { return A.Length; }
    }
    public Matrix(StreamReader sr):
this(Int16.Parse(sr.ReadLine()),Int16.Parse(sr.ReadLine()),sr.ReadLine())
    {       string str;
            transpon=false;
            int i=0,j=0,k=0,newk=0;
            if(!econom)
            {       while (((str=sr.ReadLine())!=null)&&(i<N))
                    {       j=0;
                            k=0;
                            newk=0;
                            do
                            {       newk=str.IndexOf('\t',k+1);
                                    this[i,j]=double.Parse(newk>0?str.Substring(k,
                                    newk-k):str.Substring(k));
                                    k=newk;
                                    j++;
                            }while(k!=-1);
                            i++;
                    }
            }
```

```
		else
		{	fiveDiag = true;
			if ((str=sr.ReadLine())!=null)
			{
				j=0;
				k=0;
				newk=0;
				do
				{	newk=str.IndexOf(' ',k+1);
					A[j]=double.Parse(newk>0?str.Substring(k,
					newk-k):str.Substring(k));
					k=newk;
					j++;
				}while((k!=-1)&&(j<N));
			}
			else throw new ArrayTypeMismatchException();
			if ((str=sr.ReadLine())!=null)
			{
				j=0;
				k=0;
				newk=0;
				do
				{	newk=str.IndexOf(' ',k+1);
					B[j]=double.Parse(newk>0?str.Substring(k,
					newk-k):str.Substring(k));
					k=newk;
					j++;
				}while((k!=-1)&&(j<N-1));
			}
			else throw new ArrayTypeMismatchException();
			if ((str=sr.ReadLine())!=null)
			{
				j=0;
				k=0;
				newk=0;
				do
				{	newk=str.IndexOf(' ',k+1);
					C[j]=double.Parse(newk>0?str.Substring(k,
					newk-k):str.Substring(k));
					k=newk;
					j++;
				}while((k!=-1)&&(j<N-Nx));
			}
			else throw new ArrayTypeMismatchException();
		}
	}

	public double this[int i, int j]
	{
		get {return GetElement(i, j);}
		set {SetElement(i, j, value);}
	}

	public double GetElement(int i, int j)
	{	if(transpon)
		{	int temp=i;
			i=j;
			j=temp;
		}
		if ((i < 0) || (j < 0) || (i >= N) || (j >= N))// Выход за границы
			throw new ArgumentOutOfRangeException();
		if(econom)
		{	if (fiveDiag)  // если 5-диагональная матрица
			{	if (i==j)// Диагональ
					return A[i];
				if(i>j)// Нижний треугольник
				{
					if(i==j+1) return B[j];
					if(i==j+Nx) return C[j];
				}
				else// Верхний треугольник
				{
					if(i==j-1) return B[i];
					if(i==j-Nx) return C[i];
				}
			}
			else	//нижняя треугольная трехдиагональная матрица
			{
				if (i==j)// Диагональ
					return A[i];
```

```
                        // Нижний треугольник
                        if (i > j)
                        {
                                if(i==j+1) return B[j];
                                if(i==j+Nx) return C[j];
                        }

                }
                return 0;
        }
        else return M[i][j];
}
public void SetElement(int i, int j, double value)
{       if(transpon)
        {       int temp=i;
                i=j;
                j=temp;
        }
        if ((i < 0) || (j < 0) || (i >= N) || (j >= N))// Выход за границы
                throw new ArgumentOutOfRangeException();
        if(econom)
        {       if (fiveDiag)
                {       if (i==j)// Диагональ
                                A[i]=value;
                        if(i>j)// Нижний треугольник
                        {
                                if(i==j+1) B[j]=value;
                                if(i==j+Nx) C[j]=value;
                        }
                        else// Верхний треугольник
                        {
                                if(i==j-1) B[i]=value;
                                if(i==j-Nx) C[i]=value;
                        }
                }
                else
                {       if (i==j)// Диагональ
                                A[i]=value;
                        // Нижний треугольник

                        if((i==j+1)&&(i>j)) B[j]=value;
                        if((i==j+Nx)&&(i>j)) C[j]=value;
                        if((i==j-1)&&(i<j)) B[i]=value;
                        if((i==j-Nx)&&(i<j)) C[i]=value;

                }
        }
        else M[i][j]=value;
}
public static Matrix operator *(Matrix m1, Matrix m2)
        // умножение матрицы на матрицу
{       Matrix res = null;
        int len=0;
        if ((len=m1.N) != m2.N) throw new ArgumentOutOfRangeException();
        else
        {       res = new Matrix(len,m1.Nx,false,true);
                for (int n = 0; n < len; n++)
                {       for (int i=0; i<len; i++)
                        {       res[n,i]=0;
                                for (int j = 0; j < len; j++)
                                {res[n,i] += m1[n, j]*m2[j,i];}
                        }
                }
        }
        return res;
}
public override string ToString()
{       string str="";
        for(int i=0; i<N;i++)
        {for(int j=0; j<N; j++) str+=this[i,j].ToString("#,##0.00")+
                        ((j<N-1)?"\t":"");//"#,###0.000"
                str+='\n';
        }
        return str;
}
}
public class Vector
{       public double[] V;
        public Vector(int size)
        {
```

```
        V=new double[size];
}
public Vector(int size, double[] V)
{
        this.V=new double[size];
        for(int i=0;i<size;i++)
                this.V[i]=V[i];
}
public int GetLenght
{       get { return V.Length; }
}
public Vector(StreamReader sr):this(Int16.Parse(sr.ReadLine()))
{       string str;
        int j=0,k=0,newk=0;
        if ((str=sr.ReadLine())!=null)
        {       j=0;
                k=0;
                newk=0;
                do
                {       newk=str.IndexOf(' ',k+1);
                        V[j]=double.Parse(newk>0?str.Substring(k,newk-k):str.Substring(k));
                        k=newk;
                        j++;
                }while((k!=-1)&&(j<V.Length));
        }
        else throw new ArrayTypeMismatchException();
}

public double this[int i]
{       get
        {       if(i>=V.Length) throw new ArgumentOutOfRangeException();
                return V[i];
        }
        set
        {       if(i>=V.Length) throw new ArgumentOutOfRangeException();
                V[i]=value;
        }
}
public static double operator *(Vector v1, Vector v2)
        //умножение вектора на вектор
{       double res = 0;
        int len=0;
        if ((len=v1.V.Length) != v2.V.Length) throw new ArgumentOutOfRangeException();
        else
                for (int i = 0; i < len; i++)
                {       res += v1[i]*v2[i];
                }
        return res;
}
public static Vector operator +(Vector v1, Vector v2)
        // сумма двух векторов
{       Vector res = null;
        int len=0;
        if ((len=v1.V.Length) != v2.V.Length) throw new ArgumentOutOfRangeException();
        else
        {       res = new Vector(len);
                for (int i = 0; i < len; i++)
                {res[i] = v1[i] + v2[i];
                }
        }
        return res;
}
public static Vector operator -(Vector v1, Vector v2)
        // разность двух векторов
{
        Vector res = null;
        int len=0;
        if ((len=v1.V.Length) != v2.V.Length) throw new ArgumentOutOfRangeException();
        else
        {
                res = new Vector(len);
                for (int i = 0; i < len; i++)
                {
                        res[i] = v1[i] - v2[i];
                }
        }
        return res;
}
public static Vector operator *(Matrix m, Vector v)
        // умножение матрицы на вектор
```

```
		{
			Vector res = null;
			int len=0;
			if ((len=m.N) != v.V.Length) throw new ArgumentOutOfRangeException();
			else
			{
				res = new Vector(len);
				for (int i = 0; i < len; i++)
				{
					res[i] = 0;
					for (int j = 0; j < len; j++)
					{res[i] += m[i, j]*v[j];
					}
				}
			}
			return res;
		}
		public static Vector operator *(Vector v, Matrix m)
			// умножение вектора на матрицу
		{	Vector res = null;
			int len=0;
			if ((len=m.N) != v.V.Length) throw new ArgumentOutOfRangeException();
			else
			{
				res = new Vector(len);
				for (int j = 0; j < len; j++)
				{	res[j] = 0;
					for (int i = 0; i < len; i++)
					{res[j] += m[i, j]*v[i];
					}
				}
			}
			return res;
		}
		public static Vector operator *(double d, Vector v)
			//умножение скаляра на вектор
		{	int len=v.V.Length;
			Vector res = new Vector(len);
			for (int i = 0; i < len; i++)
			{res[i] = v[i]*d;
			}
			return res;
		}
		public static Vector operator *(Vector v, double d)
			//умножение вектора на скаляр
		{	int len=v.V.Length;
			Vector res = new Vector(len);
			for (int i = 0; i < len; i++)
			{res[i] = v[i]*d;
			}
			return res;
		}
		public static double Abs(Vector v)
		{return (Math.Sqrt((v*v)));
		}

		public override string ToString()
		{	string str="";
			int len=V.Length;
			for(int i=0;i<len;i++) str+=V[i].ToString("#,#####0.00000")+" ";
			return str;
		}
	}
}
```

Метод сопряженных градиентов

```
using System;
using System.Windows.Forms;

namespace MMKursProject
{
	public class Solving
	{
		public static Vector X;
		public static int Nx = 4;
		public static int Ny = 4;
		public static double Eps=0.00000001;
		public static double xLeft=0, xRight=1, yLeft=0, yRight=1;
```

```
public static double hx, hy;
public static int N, i, j;
public static double kappa1 = 1;
public static double kappa2 = 1;
public static double kappa3 = 1;
public static double kappa4 = 1;
public static double[] xi, yj;
public static double[] Ck, Bk, Ak, Fk,Uk;
public static int k = 0,iter=0;
public static double maxerror;
public static DateTime start;
public static TimeSpan ts;
public static int State=0;

public static void Solve()
{
        try
        {       start = DateTime.Now;
                k=0;
                State=1;
                N = Nx*(Ny+1);
                hx = (xRight - xLeft)/Nx;
                hy = (yRight - yLeft)/Ny;
                xi = new double[Nx + 1];
                yj = new double[Ny + 1];

                // сетка по x
                for (i = 0; i <= Nx; i++)
                {       xi[i] = xLeft + i*hx;         }

                // сетка по y
                for (j = 0; j <= Ny; j++)
                {       yj[j] = yLeft + j*hy;         }

                Ck = new double[N];
                Bk = new double[N];
                Ak = new double[N];
                Fk = new double[N];
                Uk = new double[N];
                for (j=0; j<=Ny; j++)
                        for (i=1; i<=Nx; i++)
                        {       k=j*Nx+i-1;
                                Uk[k]=u(xi[i],yj[j]);
                        }
                for (i=1;i<=Nx-1;i++)
                        for (j=1;j<=Ny-1;j++)
                        {
                                k=j*Nx+i-1;
Ak[k]=(hy/hx)*(k1(xi[i],xi[i+1],yj[j])+k1(xi[i],xi[i-
1],yj[j]))+(hx/hy)*(k2(xi[i],yj[j],yj[j+1])+k2(xi[i],yj[j],yj[j-1]));
Bk[k]=-(hy/hx)*(k1(xi[i],xi[i+1],yj[j]));
Ck[k]=-(hx/hy)*(k2(xi[i],yj[j],yj[j+1]));
Fk[k]=(hx*hy)*fun(xi[i],yj[j]);
if(i==1) Fk[k]+=(hy/hx)*V1(yj[j])*k1(xi[i],xi[i-1],yj[j]);
}
                for (j=1;j<=Ny-1;j++)
                {       i=Nx;
                        k=j*Nx+i-1;
Ak[k]=hy*(kappa2+(k1(xi[i],xi[i-
1],yj[j]))/hx)+(hx/(2*hy))*(k2(xi[i],yj[j],yj[j+1])+k2(xi[i],yj[j],yj[j-1]));
Ck[k]=-(hx/(2*hy))*k2(xi[i],yj[j],yj[j+1]);
Fk[k]=((hx/2)*hy)*fun(xi[i],yj[j]) + hy*V2(yj[j]);
}
                for (i=1;i<=Nx-1;i++)
                {       j=Ny;
                        k=j*Nx+i-1;
                Ak[k]=(hy/(2*hx))*(k1(xi[i],xi[i+1],yj[j])+k1(xi[i],xi[i-1],yj[j])) +
        hx*(k2(xi[i],yj[j],yj[j-1])/hy + kappa4);
        Bk[k]=-(hy/(2*hx))*k1(xi[i],xi[i+1],yj[j]);
        Fk[k]=((hy/2)*hx)*fun(xi[i],yj[j]) + hx*V4(xi[i]);
        if (i==1) Fk[k]+=(hy/(2*hx))*V1(yj[j])*k1(xi[i],xi[i-1],yj[j]);
        }
                for (i=1;i<=Nx-1;i++)
                {       j=0;
                        k=j*Nx+i-1;
Ak[k]=(hy/(2*hx))*(k1(xi[i],xi[i+1],yj[j])+k1(xi[i],xi[i-1],yj[j])) +
hx*(k2(xi[i],yj[j],yj[j+1])/hy + kappa3);
Bk[k]=-(hy/(2*hx))*k1(xi[i],xi[i+1],yj[j]);
Ck[k]=-(hx/hy)*(k2(xi[i],yj[j],yj[j+1]));
Fk[k]=((hy/2)*hx)*fun(xi[i],yj[j]) + hx*V3(xi[i]);
```

```
            if (i==1) Fk[k]+=(hy/(2*hx))*V1(yj[j])*k1(xi[i],xi[i-1],yj[j]);
            }

Ak[Nx-1]=(hy/2)*(kappa2 + k1(xi[Nx],xi[Nx-1],yj[0])/hx) + (hx/2)*(kappa3 + k2(xi[Nx],yj[0],yj[1])/hy);
Ck[Nx-1]=-(hx/(2*hy))*k2(xi[Nx],yj[0],yj[1]);
Fk[Nx-1]=((hx*hy)/4)*fun(xi[Nx],yj[0]) + (hy/2)*V2(yj[0]) + (hx/2)*V3(xi[Nx]);

        Ak[Nx*Ny+Nx-1]=(hy/2)*(kappa2 + k1(xi[Nx],xi[Nx-1],yj[Ny])/hx) + (hx/2)*(kappa4 +
        k2(xi[Nx],yj[Ny],yj[Ny-1])/hy);
        Fk[Nx*Ny+Nx-1]=((hx*hy)/4)*fun(xi[Nx],yj[Ny]) + (hy/2)*V2(yj[Ny]) + (hx/2)*V4(xi[Nx]);
//---------------------------------------------------------------------------------------
                // формируем вектора для матрицы
                    Vector A = new Vector(N);
                    Vector B = new Vector(N - 1);
                    Vector C = new Vector(N - Nx);
                    Vector F = new Vector(N);
                    Vector X0 = new Vector(N);
                    Vector UU = new Vector(N);
                    for ( k = 0; k < N; k++) A[k] = Ak[k];
                    for ( k = 0; k < N-1; k++) B[k] = Bk[k];
                    for ( k = 0; k < N - Nx; k++) C[k] = Ck[k];
                    for ( k = 0; k < N; k++) F[k] = Fk[k];
                    for ( k = 0; k < N; k++) UU[k] = Uk[k];
                    // формируем матрицу
                    Matrix MM = new Matrix(N, Nx, A, B, C);
                    MM.fiveDiag = true;
                    //X = Metod(MM, F, X0);
                    X = MetodHoleckogo(MM, F, X0);
                    maxerror=Math.Abs(UU[0]-X[0]);
                    for (i=1;i<N;i++)
                    {
                        if(maxerror<Math.Abs(UU[i]-X[i]))
                        {
                            maxerror=Math.Abs(UU[i]-X[i]);
                        }
                    }
                    DateTime end = DateTime.Now;
                    ts=end-start;
                    State=2;
                }
                catch (Exception exc)
                {
                    MessageBox.Show(exc.Message);
                }
            }

            // метод сопряженных градиентов без предобуславливания
            public static Vector Metod(Matrix A, Vector F, Vector x0)
            {       iter = 0;
                bool Stop = false;
                Vector r0 = new Vector(F.GetLenght);
                Vector r1 = new Vector(F.GetLenght);
                Vector S1 = new Vector(F.GetLenght);
                Vector x = new Vector(F.GetLenght);
                Vector x00 = new Vector(F.GetLenght);
                double alfa, betta;
                // с чем начинаем решать
                x00 = x0;
                iter = 0;
                // невязка
                r0 = (F - A*x0);
                //  вспомогательный вектор в методе
                S1 = r0;
                //проверка вектора A*s1
                while (!Stop)
                {
                    iter++;
                    // подсчет альфы
                    alfa = (r0*r0)/((A*S1)*S1);
                    x = x0 + alfa*S1;
                    r1 = r0 - alfa*(A*S1);
                    // проверка на сходимость - завершение метода
                    if ((Vector.Abs(r1)/Vector.Abs(F)) < Eps)
                    {
                        if ((Vector.Abs(F - A*x00)/Vector.Abs(F)) > Eps)
                        {
                            Stop = true;
                        }
                        else return x;
                    }
```

```
                betta = (r1*r1)/(r0*r0);
                S1 = r1 + betta*S1;
                r0 = r1;
                x0 = x;
        }
        return x;
}
// метод сопряженных градиентов с предобуславливанием
public static Vector MetodHoleckogo(Matrix A, Vector F, Vector x0)
{
        iter = 0;

        bool Stop = false;
        Vector r0 = new Vector(F.GetLenght);
        Vector r1 = new Vector(F.GetLenght);
        Vector S1 = new Vector(F.GetLenght);
        Vector x = new Vector(F.GetLenght);
        Vector x00 = new Vector(F.GetLenght);
        Vector w0 = new Vector(F.GetLenght);
        Vector w1 = new Vector(F.GetLenght);
        Matrix L;
        double alfa, betta;
        // с чем начинаем решать
        x00 = x0;
        iter = 0;
        // невязка
        r0 = (F - A*x0);
        L = CreateMatrixL(A);
        w0 = FindW(L, r0);
        //  вспомогательный вектор в методе
        S1 = w0;
        //проверка вектора A*s1
        while (!Stop)
        {
                iter++;
                // подсчет альфы
                alfa = (w0*r0)/((A*S1)*S1);
                x = x0 + alfa*S1;
                r1 = r0 - alfa*(A*S1);
                // проверка на сходимость - завершение метода
                if ((Vector.Abs(r1)/Vector.Abs(F)) < Eps)
                {
                        if ((Vector.Abs(F - A*x00)/Vector.Abs(F)) > Eps)
                        {
                                Stop = true;
                        }
                        else return x;
                }
                w1 = FindW(L, r1);
                betta = (w1*r1)/(w0*r0);
                S1 = w1 + betta*S1;
                r0 = r1;
                w0 = w1;
                x0 = x;
        }
        return x;
}
public static Matrix CreateMatrixL(Matrix M)
{
        //Получаем вектора a,b,c из основной матрицы для формирования векторов в матрицу L
        Vector A = new Vector(M.GetLength);
        Vector B = new Vector(M.GetLength);
        Vector C = new Vector(M.GetLength);
        double[] Al = new double[M.GetLength];
        double[] Bl = new double[M.GetLength];
        double[] Cl = new double[M.GetLength];
        for (int i = 0; i < M.N; i++)
        {
                for (int j = 0; j < M.N; j++)
                {
                        if (i == j) // Диагональ
                                A[i] = M[i, j];
                        if (i > j) // Нижний треугольник
                        {
                                if (i == j + 1) B[j] = M[i, j];
                                if (i == j + M.Nx) C[j] = M[i, j];
                        }
                }
        }
        Al[0]=Math.Sqrt(A[0]);
```

```
            Bl[0]=B[0]/Al[0];
            Cl[0] = C[0]/Al[0];
            for (int i = 1; i < M.Nx; i++)
            {
                    Al[i] = Math.Sqrt(A[i]-Bl[i-1]*Bl[i-1]);
                    Bl[i] = B[i]/Al[i];
                    Cl[i] = C[i]/Al[i];
            }

            for (int i = M.Nx; i < M.N; i++)
            {
                    Al[i] = Math.Sqrt(A[i]-Bl[i-1]*Bl[i-1]-Cl[i-M.Nx]*Cl[i-M.Nx]);
                    Bl[i] = B[i]/Al[i];
                    Cl[i] = C[i]/Al[i];
            }
            //формируем вектора для построения матрицы
            A = new Vector(M.N);
            B = new Vector(M.N - 1);
            C = new Vector(M.N - M.Nx);
            for (int i = 0; i < M.N; i++)
                    A[i] = Al[i];
            for (int i = 0; i < M.N - 1; i++)
                    B[i] = Bl[i];
            for (int i = 0; i < M.N - M.Nx; i++)
            {
                    C[i] = Cl[i];
            }

            Matrix L = new Matrix(M.N, M.Nx, A, B, C);
            return L;
    }
    // метод Гаусса - решает систему вниз
    public static Vector GaussDown(Matrix M, Vector F)
    {
            Vector x = new Vector(M.N);
            x[0] = F[0]/M.GetElement(0, 0);
for (int i = 1; i < M.Nx; i++) x[i] = (F[i] - M.GetElement(i, i - 1)*x[i - 1])/M.GetElement(i, i);

            for (int i = M.Nx; i < M.N; i++)
            x[i] = (F[i] - M.GetElement(i, i - M.Nx)*x[i - M.Nx] - M.GetElement(i, i - 1)*x[i -
            1])/M.GetElement(i, i);
            return x;
    }
    // метод Гаусса - решает систему вверх
    public static Vector GaussUp(Matrix M, Vector F)
    {
            M.transpon = true;
            Vector x = new Vector(M.N);
            x[M.N - 1] = F[M.N - 1]/M.GetElement(M.N - 1, M.N - 1);
            for (int i = M.N - 2; i >= (M.N - M.Nx); i--)
                    x[i] = (F[i] - M.GetElement(i, i + 1)*x[i + 1])/M.GetElement(i, i);

            for (int i = M.N - M.Nx - 1; i >= 0; i--)
    x[i] = (F[i] - M.GetElement(i, i + M.Nx)*x[i + M.Nx] - M.GetElement(i, i + 1)*x[i +
    1])/M.GetElement(i, i);
            M.transpon = false;
            return x;
    }
    // нахождение дополнительного вектора w
    public static Vector FindW(Matrix L, Vector r)
    {
            Vector temp = new Vector(L.GetLength, GaussDown(L, r).V);
            return GaussUp(L, temp);
    }

    // функция к1
    public static double k1(double xi, double xi_1, double yi)
    {
            double x = (xi + xi_1)/2;
            double res;
            //res = 2;//test 1
            //res = 2*x + yi + 1;//test 2
            res = 1 + 2*x*x + yi;
            return res;

    }
    // функция к2
    public static double k2(double xi, double yi, double yi_1)
    {
            double res;
```

```
            double y = (yi + yi_1)/2;
            //res = 5;//test 1
            //res = xi + 2*y + 1;//test2;
            res = 1 + xi + 2*y*y;
            return res;
        }
        //решение уравнения - u
        public static double u(double x, double y)
        {
            double res;
            //res = 3;//test1
            //res = 2*x+3*y+5;//test2
            res = 2*x*x + 3*y*y*y + 4;
            return res;
        }

        public static double V1(double y)
        {
            double res;
            //res = 3;//test1
            //res = 2*xLeft+3*y+5;//test2
            res = 2*xLeft*xLeft+3*y*y*y + 4;
            return res;
        }

        public static double V2(double y)
        {
            double res;
            //res = 3;//test1
            //res = kappa2*(2*xRight+3*y+5)+(2*xRight+y+1)*2;//test2
            res = kappa2*(2*xRight*xRight+3*y*y*y+4)+(1+2*xRight*xRight+y)*4*xRight;
            return res;
        }

        public static double V3(double x)
        {
            double res;
            //res = 3;//test 1
            //res = kappa3*(2*x+3*yLeft+5)-(x+2*yLeft+1)*3;//test2
            res = kappa3*(2*x*x+4+3*yLeft*yLeft*yLeft)-(1+x+2*yLeft*yLeft)*9*yLeft*yLeft;
            return res;
        }

        public static double V4(double x)
        {
            double res;
            //res = 3;//test1
            //res = kappa4*(2*x+3*yRight+5)+(x+2*yRight+1)*3;//test2
            res = kappa4*(2*x*x+4+3*yRight*yRight*yRight)+(1+x+2*yRight*yRight)*9*yRight*yRight;
            return res;
        }
        // наш функционал - правая часть уравнения
        public static double fun(double x, double y)
        {
            double res;
            //res = 0;//test 1;
            //res = -10;//test2
            res = -(24*x*x + 18*x*y + 22*y + 72*y*y*y + 4);
            return res;
        }
    }
}
```

Приложение 2

Анализ порядка аппроксимации уравнения и граничных условий, вывод выражения для главного члена погрешности аппроксимации:

Анализ порядка аппроксимации уравнения и вывод выражения для главного члена погрешности аппроксимации.

$$-\left[\frac{\partial}{\partial x}\left(k1(x,y)\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(k2(x,y)\frac{\partial u}{\partial y}\right)\right]=f(x,y),$$

$$-\left[\hbar_j\left(\frac{k1_{i+1/2,j}(V_{i+1,j}-V_{i,j})}{h_{i+1}}-\frac{k1_{i-1/2,j}(V_{i,j}-V_{i-1,j})}{h_i}\right)+\hbar_i\left(\frac{k2_{i,j+1/2}(V_{i,j+1}-V_{i,j})}{h_{j+1}}-\frac{k2_{i,j-1/2}(V_{i,j}-V_{i-1,j})}{h_j}\right)\right]=$$

$$=\hbar_i\hbar_j f_{i,j}$$

$$h=\hbar_i=\hbar_j=h_i=h_j$$

$$\psi_{ij}=hf_{i,j}+\left[\hbar_j\left(\frac{k1_{i+1/2,j}(V_{i+1,j}-V_{i,j})}{h}-\frac{k1_{i-1/2,j}(V_{i,j}-V_{i-1,j})}{h}\right)+\hbar_i\left(\frac{k2_{i,j+1/2}(V_{i,j+1}-V_{i,j})}{h}-\frac{k2_{i,j-1/2}(V_{i,j}-V_{i-1,j})}{h}\right)\right]$$

$$V_{i+1,j}=V(x_i+h,y_j)=\left[V+h\frac{\partial V}{\partial x}+\frac{h^2}{2}\frac{\partial^2 V}{\partial x^2}+\frac{h^3}{6}\frac{\partial^3 V}{\partial x^3}+\ldots\right]_{\substack{x=x_i\\y=y_j}}$$

$$V_{i-1,j}=V(x_i-h,y_j)=\left[V-h\frac{\partial V}{\partial x}+\frac{h^2}{2}\frac{\partial^2 V}{\partial x^2}-\frac{h^3}{6}\frac{\partial^3 V}{\partial x^3}+\ldots\right]_{\substack{x=x_i\\y=y_j}}$$

$$V_{i,j+1}=V(x_i,y_j+h)=\left[V+h\frac{\partial V}{\partial y}+\frac{h^2}{2}\frac{\partial^2 V}{\partial y^2}+\frac{h^3}{6}\frac{\partial^3 V}{\partial y^3}+\ldots\right]_{\substack{x=x_i\\y=y_j}}$$

$$V_{i,j-1}=V(x_i,y_j-h)=\left[V-h\frac{\partial V}{\partial y}+\frac{h^2}{2}\frac{\partial^2 V}{\partial y^2}-\frac{h^3}{6}\frac{\partial^3 V}{\partial y^3}+\ldots\right]_{\substack{x=x_i\\y=y_j}}$$

$$\frac{V_{i+1,j}-V_{i,j}}{h}=\left[\frac{\partial V}{\partial x}+\frac{h}{2}\frac{\partial^2 V}{\partial x^2}+\frac{h^2}{6}\frac{\partial^3 V}{\partial x^3}+\ldots\right]_{\substack{x=x_i\\y=y_j}}$$

$$\frac{V_{i,j}-V_{i-1,j}}{h}=\left[\frac{\partial V}{\partial x}-\frac{h}{2}\frac{\partial^2 V}{\partial x^2}+\frac{h^2}{6}\frac{\partial^3 V}{\partial x^3}+\ldots\right]_{\substack{x=x_i\\y=y_j}}$$

$$\frac{V_{i,j+1}-V_{i,j}}{h}=\left[\frac{\partial V}{\partial y}+\frac{h}{2}\frac{\partial^2 V}{\partial y^2}+\frac{h^2}{6}\frac{\partial^3 V}{\partial y^3}+\ldots\right]_{\substack{x=x_i\\y=y_j}}$$

$$\frac{V_{i,j}-V_{i,j-1}}{h}=\left[\frac{\partial V}{\partial y}-\frac{h}{2}\frac{\partial^2 V}{\partial y^2}+\frac{h^2}{6}\frac{\partial^3 V}{\partial y^3}+\ldots\right]_{\substack{x=x_i\\y=y_j}}$$

$$k1_{i+1/2,j} = k1(x_i + \frac{h}{2}, y_j) = \left[k1 + \frac{h}{2}\frac{\partial k1}{\partial x} + \frac{h^2}{8}\frac{\partial^2 k1}{\partial x^2} + \frac{h^3}{48}\frac{\partial^3 k1}{\partial x^3} + \ldots \right]_{\substack{x=x_i \\ y=y_j}}$$

$$k1_{i-1/2,j} = k1(x_i - \frac{h}{2}, y_j) = \left[k1 - \frac{h}{2}\frac{\partial k1}{\partial x} + \frac{h^2}{8}\frac{\partial^2 k1}{\partial x^2} - \frac{h^3}{48}\frac{\partial^3 k1}{\partial x^3} + \ldots \right]_{\substack{x=x_i \\ y=y_j}}$$

$$k2_{i,j+1/2} = k2(x_i, y_j + \frac{h}{2}) = \left[k2 + \frac{h}{2}\frac{\partial k2}{\partial y} + \frac{h^2}{8}\frac{\partial^2 k2}{\partial y^2} + \frac{h^3}{48}\frac{\partial^3 k2}{\partial y^3} + \ldots \right]_{\substack{x=x_i \\ y=y_j}}$$

$$k2_{i,j-1/2} = k2(x_i, y_j - \frac{h}{2}) = \left[k2 - \frac{h}{2}\frac{\partial k2}{\partial y} + \frac{h^2}{8}\frac{\partial^2 k2}{\partial y^2} - \frac{h^3}{48}\frac{\partial^3 k2}{\partial y^3} + \ldots \right]_{\substack{x=x_i \\ y=y_j}}$$

$$\frac{k1_{i+1/2,j}(V_{i+1,j} - V_{i,j})}{h} = k1_{i,j}\frac{\partial V}{\partial x} + h\left[\frac{1}{2}k1\frac{\partial^2 V}{\partial x^2} + \frac{1}{2}\frac{\partial k1}{\partial x}\frac{\partial V}{\partial x} \right]_{\substack{x=x_i \\ y=y_j}} +$$

$$+h^2\left[\frac{1}{6}k1\frac{\partial^3 V}{\partial x^3} + \frac{1}{4}\frac{\partial k1}{\partial x}\frac{\partial^2 V}{\partial x^2} + \frac{1}{8}\frac{\partial^2 k1}{\partial x^2}\frac{\partial V}{\partial x} \right]_{\substack{x=x_i \\ y=y_j}} +$$

$$+h^3\left[\frac{1}{24}k1\frac{\partial^4 V}{\partial x^4} + \frac{1}{12}\frac{\partial k1}{\partial x}\frac{\partial^3 V}{\partial x^3} + \frac{1}{16}\frac{\partial^2 k1}{\partial x^2}\frac{\partial^2 V}{\partial x^2} + \frac{1}{48}\frac{\partial^3 k1}{\partial x^3}\frac{\partial V}{\partial x} \right]_{\substack{x=x_i \\ y=y_j}} + \ldots$$

$$\frac{k1_{i-1/2,j}(V_{i,j} - V_{i-1,j})}{h} = k1_{i,j}\frac{\partial V}{\partial x} - h\left[\frac{1}{2}k1\frac{\partial^2 V}{\partial x^2} + \frac{1}{2}\frac{\partial k1}{\partial x}\frac{\partial V}{\partial x} \right]_{\substack{x=x_i \\ y=y_j}} +$$

$$+h^2\left[\frac{1}{6}k1\frac{\partial^3 V}{\partial x^3} + \frac{1}{4}\frac{\partial k1}{\partial x}\frac{\partial^2 V}{\partial x^2} + \frac{1}{8}\frac{\partial^2 k1}{\partial x^2}\frac{\partial V}{\partial x} \right]_{\substack{x=x_i \\ y=y_j}} -$$

$$-h^3\left[\frac{1}{24}k1\frac{\partial^4 V}{\partial x^4} + \frac{1}{12}\frac{\partial k1}{\partial x}\frac{\partial^3 V}{\partial x^3} + \frac{1}{16}\frac{\partial^2 k1}{\partial x^2}\frac{\partial^2 V}{\partial x^2} + \frac{1}{48}\frac{\partial^3 k1}{\partial x^3}\frac{\partial V}{\partial x} \right]_{\substack{x=x_i \\ y=y_j}} + \ldots$$

$$\frac{k2_{i,j+1/2}(V_{i,j+1} - V_{i,j})}{h} = k2_{i,j}\frac{\partial V}{\partial x} + h\left[\frac{1}{2}k2\frac{\partial^2 V}{\partial x^2} + \frac{1}{2}\frac{\partial k2}{\partial x}\frac{\partial V}{\partial x} \right]_{\substack{x=x_i \\ y=y_j}} +$$

$$+h^2\left[\frac{1}{6}k2\frac{\partial^3 V}{\partial x^3} + \frac{1}{4}\frac{\partial k2}{\partial x}\frac{\partial^2 V}{\partial x^2} + \frac{1}{8}\frac{\partial^2 k2}{\partial x^2}\frac{\partial V}{\partial x} \right]_{\substack{x=x_i \\ y=y_j}} +$$

$$+h^3\left[\frac{1}{24}k2\frac{\partial^4 V}{\partial x^4} + \frac{1}{12}\frac{\partial k2}{\partial x}\frac{\partial^3 V}{\partial x^3} + \frac{1}{16}\frac{\partial^2 k2}{\partial x^2}\frac{\partial^2 V}{\partial x^2} + \frac{1}{48}\frac{\partial^3 k2}{\partial x^3}\frac{\partial V}{\partial x} \right]_{\substack{x=x_i \\ y=y_j}} + \ldots$$

$$\frac{k2_{i,j-1/2}(V_{i,j} - V_{i-1,j})}{h} = k2_{i,j}\frac{\partial V}{\partial x} - h\left[\frac{1}{2}k2\frac{\partial^2 V}{\partial x^2} + \frac{1}{2}\frac{\partial k2}{\partial x}\frac{\partial V}{\partial x} \right]_{\substack{x=x_i \\ y=y_j}} +$$

$$+h^2\left[\frac{1}{6}k2\frac{\partial^3 V}{\partial x^3} + \frac{1}{4}\frac{\partial k2}{\partial x}\frac{\partial^2 V}{\partial x^2} + \frac{1}{8}\frac{\partial^2 k2}{\partial x^2}\frac{\partial V}{\partial x} \right]_{\substack{x=x_i \\ y=y_j}} -$$

$$-h^3\left[\frac{1}{24}k2\frac{\partial^4 V}{\partial x^4} + \frac{1}{12}\frac{\partial k2}{\partial x}\frac{\partial^3 V}{\partial x^3} + \frac{1}{16}\frac{\partial^2 k2}{\partial x^2}\frac{\partial^2 V}{\partial x^2} + \frac{1}{48}\frac{\partial^3 k2}{\partial x^3}\frac{\partial V}{\partial x} \right]_{\substack{x=x_i \\ y=y_j}} + \ldots$$

$$\psi_{i,j}=hf_{i,j}+h\left[k1\frac{\partial^2V}{\partial x^2}+\frac{\partial k1}{\partial x}\frac{\partial V}{\partial x}\right]_{\substack{x=x_i\\y=y_j}}+h^3\left[\frac{1}{12}k1\frac{\partial^4V}{\partial x^4}+\frac{1}{6}\frac{\partial k1}{\partial x}\frac{\partial^3V}{\partial x^3}+\frac{1}{8}\frac{\partial^2k1}{\partial x^2}\frac{\partial^2V}{\partial x^2}+\frac{1}{24}\frac{\partial^3k1}{\partial x^3}\frac{\partial V}{\partial x}\right]_{\substack{x=x_i\\y=y_j}}+$$

$$+h\left[k2\frac{\partial^2V}{\partial y^2}+\frac{\partial k2}{\partial y}\frac{\partial V}{\partial y}\right]_{\substack{x=x_i\\y=y_j}}+h^3\left[\frac{1}{12}k2\frac{\partial^4V}{\partial y^4}+\frac{1}{6}\frac{\partial k2}{\partial y}\frac{\partial^3V}{\partial y^3}+\frac{1}{8}\frac{\partial^2k2}{\partial y^2}\frac{\partial^2V}{\partial y^2}+\frac{1}{24}\frac{\partial^3k2}{\partial y^3}\frac{\partial V}{\partial y}\right]_{\substack{x=x_i\\y=y_j}}+\ldots=$$

$$=h\left[f_{i,j}+\left(\frac{\partial}{\partial x}\left(k1\frac{\partial V}{\partial x}\right)+\frac{\partial}{\partial y}\left(k2\frac{\partial V}{\partial y}\right)\right)\right]_{\substack{x=x_i\\y=y_j}}+$$

$$+h^3\left[\left(\frac{1}{12}k1\frac{\partial^4V}{\partial x^4}+\frac{1}{6}\frac{\partial k1}{\partial x}\frac{\partial^3V}{\partial x^3}+\frac{1}{8}\frac{\partial^2k1}{\partial x^2}\frac{\partial^2V}{\partial x^2}+\frac{1}{24}\frac{\partial^3k1}{\partial x^3}\frac{\partial V}{\partial x}\right)+\right.$$

$$\left.+\left(\frac{1}{12}k2\frac{\partial^4V}{\partial y^4}+\frac{1}{6}\frac{\partial k2}{\partial y}\frac{\partial^3V}{\partial y^3}+\frac{1}{8}\frac{\partial^2k2}{\partial y^2}\frac{\partial^2V}{\partial y^2}+\frac{1}{24}\frac{\partial^3k2}{\partial y^3}\frac{\partial V}{\partial y}\right)\right]_{\substack{x=x_i\\y=y_j}}+\ldots$$

Причем, коэффициент при h^1 равен нулю – разность левой и правой частей ($(f-Lu_{ij})=0$), а коэффициент при h^3 – главный член погрешности аппроксимации

В качестве порядка погрешности аппроксимации уравнения предлагается брать разность показателей степени h при главном члене и члене, при котором стоит разность правой и левой частей.

В общем случае: $\psi_{i,j}=h^m\left[f-Lu\right]_{i,j}+h^{m+k}\Phi$ получаем порядок аппроксимации p = (m+k)-m.

Здесь p = 2.

При граничных условиях:

$$-\left[\hbar_j\left(\frac{k1_{i+1/2,j}(V_{i+1,j}-V_{i,j})}{h_{i+1}}-\frac{k1_{i-1/2,j}(V_{i,j}-V_{i-1,j})}{h_i}\right)+\hbar_i\left(\frac{k2_{i,j+1/2}(V_{i,j+1}-V_{i,j})}{h_{j+1}}-(\chi_3V_{i,j}-g3(x_i))\right)\right]=$$

$$=\hbar_i\hbar_j f_{ij},\ \hbar_j=\frac{h}{2}$$

$$\psi_{ij}=\left[\frac{1}{2}\left(\frac{k1_{i+1/2,j}(V_{i+1,j}-V_{i,j})}{h}-\frac{k1_{i-1/2,j}(V_{i,j}-V_{i-1,j})}{h}\right)+\left(\frac{k2_{i,j+1/2}(V_{i,j+1}-V_{i,j})}{h}-(\chi_3V_{i,j}-g3(x_i))\right)+\frac{h}{2}f_{ij}\right]=$$

$$=h^0\left[k2\frac{\partial V}{\partial y}-(\chi_3V-g3(x))\right]_{\substack{x=x_i\\y=y_j}}+h\frac{1}{2}\left[f+\frac{\partial}{\partial x}\left(k1\frac{\partial V}{\partial x}\right)+\frac{\partial}{\partial y}\left(k2\frac{\partial V}{\partial y}\right)\right]_{\substack{x=x_i\\y=y_j}}+$$

$$+h^2\left[\frac{1}{6}\frac{\partial k2}{\partial y}\frac{\partial^3V}{\partial y^3}+\frac{1}{4}\frac{\partial k2}{\partial y}\frac{\partial^2V}{\partial y^2}+\frac{1}{8}\frac{\partial^2k2}{\partial y^2}\frac{\partial V}{\partial y}\right]_{\substack{x=x_i\\y=y_j}}+$$

$$+h^3\frac{1}{2}\left[\frac{1}{12}k1\frac{\partial^4V}{\partial x^4}+\frac{1}{6}\frac{\partial k1}{\partial x}\frac{\partial^3V}{\partial x^3}+\frac{1}{8}\frac{\partial^2k1}{\partial x^2}\frac{\partial^2V}{\partial x^2}+\frac{1}{24}\frac{\partial^3k1}{\partial x^3}\frac{\partial V}{\partial x}\right]_{\substack{x=x_i\\y=y_j}}+\ldots$$

Коэффициент при h^0 равен нулю, т.к. он является разностью левой и правой частей гран. условия, а при h^1 стоит разность левой и правой частей уравнения.

В общем случае: $\psi_{ij}=h^l\left[l_hu-\mu\right]_{i,j}+h^{l+k}\tilde{\Phi}$ получаем порядок аппроксимации p=(l+k)-l.