|  |  |
| --- | --- |
|  | |
| Московский авиационный институт | |
| (Национальный исследовательский университет) | |
| Институт №8 «Компьютерные науки и прикладная математика» | |
| Кафедра вычислительной математики и программирования | |
|  | |
| **Лабораторные работы** | |
| по курсу «Численные методы» | |
| Вариант 1(б) | |
|  | |
|  | Выполнила: Попова Н. С. |
|  | Группа: М8О-405Б-20 |
|  | Проверил: доц. Иванов И. Э. |
|  | Дата: |
|  | Оценка: |
|  | |
| Москва, 2023 | |

# **Лабораторная работа №8**

Используя схемы переменных направлений и дробных шагов, решить двумерную начально-краевую задачу для дифференциального уравнения параболического типа. В различные моменты времени вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением U(x,y,t). Исследовать зависимость от сеточных параметров.

Аналитическое решение:

### **Теоретические сведения**

**Метод переменных направлений**

Для момента времени k+1/2 производная по иксу будет аппроксимироваться неявно, а по игреку - явно. Для момента времени k+1 наоборот.

Шаг 1.

Схема МТН для слоя :

Получим систему уравнений для всех j, чтобы получить значения в момент времени k+1/2:

*,*

*, ,* где и

Шаг 2.

Схема МТН для слоя :

Получим систему уравнений для всех i, чтобы получить значения в момент времени k+1:

*,*

*, ,* где и

**Метод дробных шагов**

Метод дробных шагов использует только неявные схемы.

Шаг 1.

Схема МДШ для слоя :

Решаем систему уравнений для всех j, чтобы получить значения в момент времени k+1/2:

*,*

*, ,* где и

Шаг 2.

Схема МДШ для слоя :

Решаем систему уравнений для всех i, чтобы получить значения в момент времени k+1:

*,*

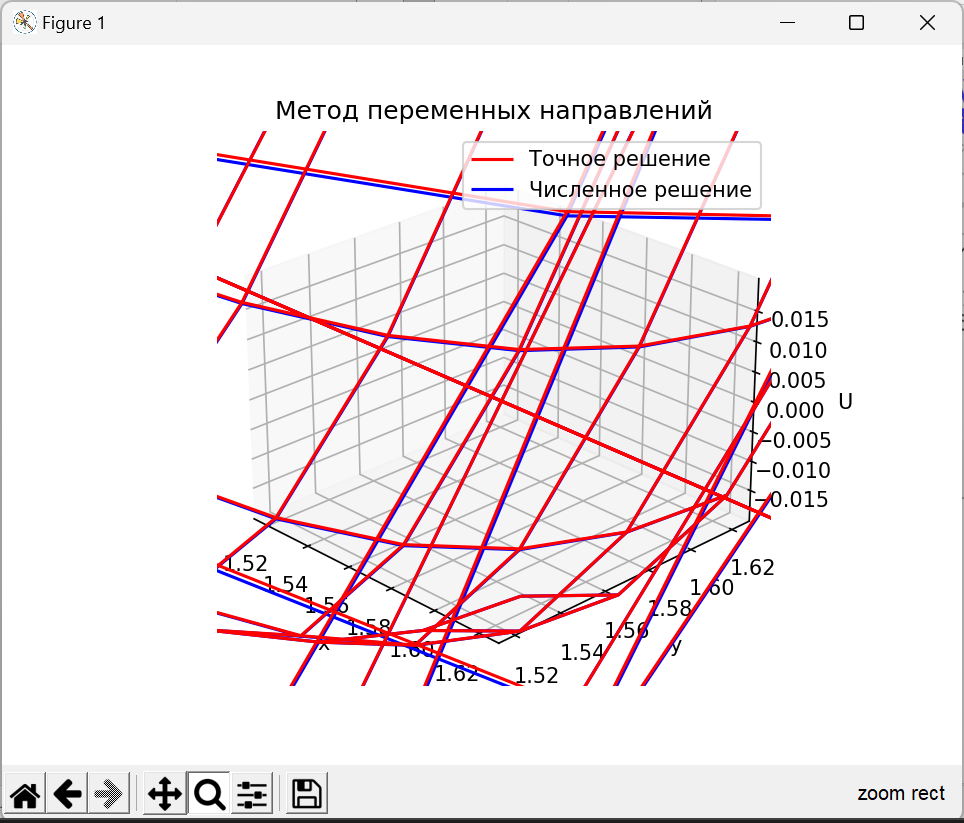
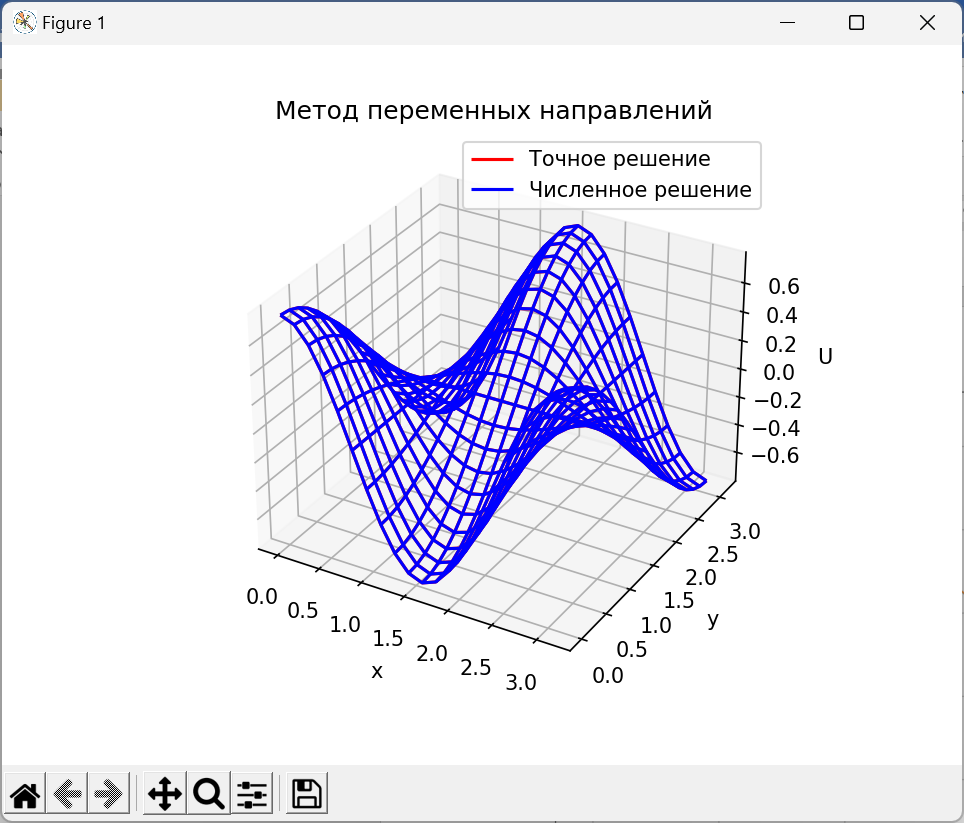
*, ,* где и

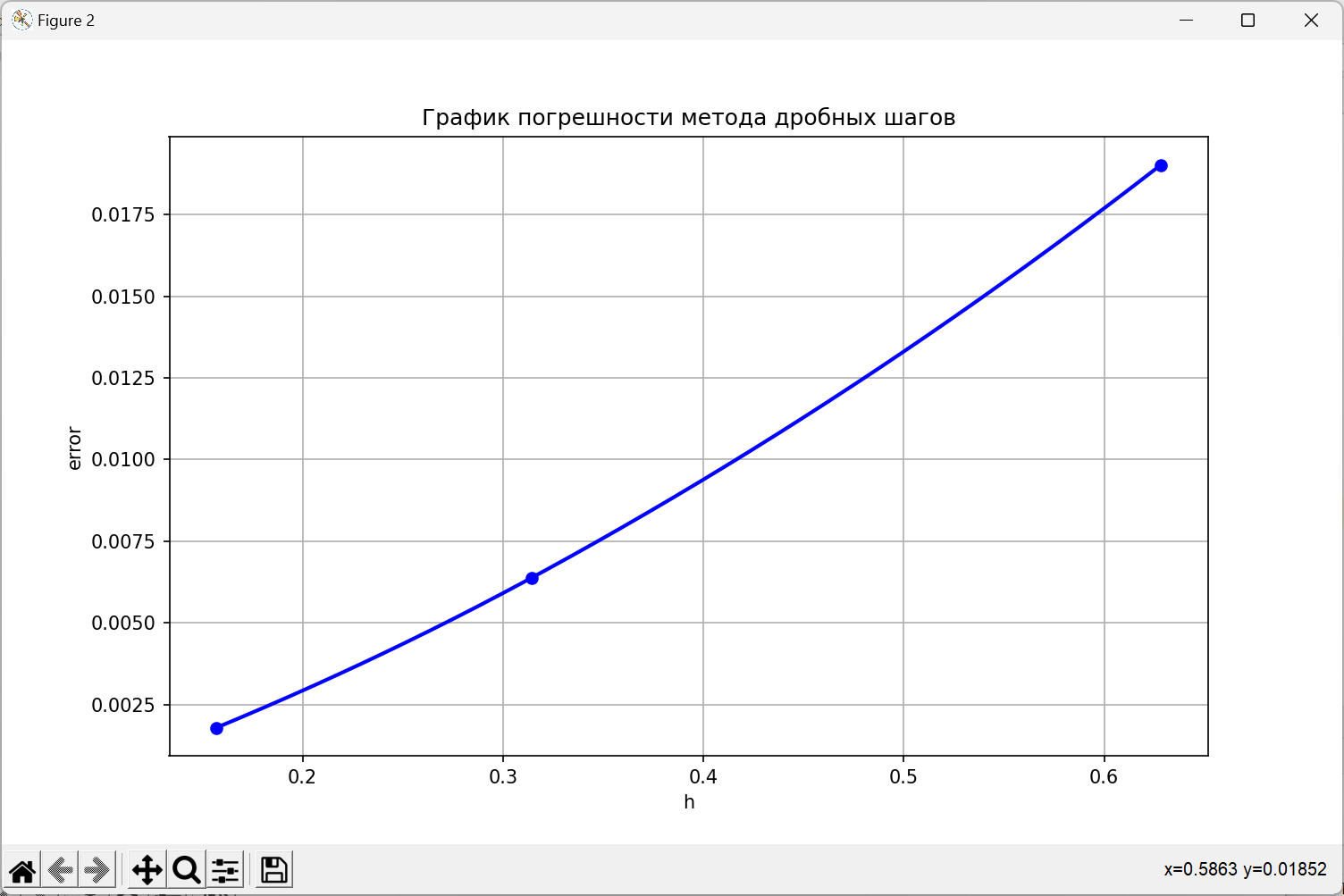
**Код программы**

**import** numpy **as** np, matplotlib.pyplot **as** plt  
**from** mpl\_toolkits.mplot3d **import** axes3d  
  
**def** analyt\_func(x, y, t):  
 **return** np.cos(2 \* x) \* np.cos(y) \* np.exp(-5 \* t)  
  
**def** phi\_0(y, t):  
 **return** np.cos(y) \* np.exp(-5 \* t)  
  
**def** phi\_1(y, t):  
 **return** np.cos(y) \* np.exp(-5 \* t)  
  
  
**def** phi\_2(x, t):  
 **return** np.cos(2 \* x) \* np.exp(-5 \* t)  
  
  
**def** phi\_3(x, t):  
 **return** -np.cos(2\* x) \* np.exp(-5 \* t)  
  
  
**def** psi(x, y):  
 **return** np.cos(2 \* x) \* np.cos(y)  
  
**def** run\_through(a, b, c, d, s):  
 P = np.zeros(s + 1)  
 Q = np.zeros(s + 1)  
  
 P[0] = -c[0] / b[0]  
 Q[0] = d[0] / b[0]  
  
 k = s - 1  
 **for** i **in** range(1, s):  
 P[i] = -c[i] / (b[i] + a[i] \* P[i - 1])  
 Q[i] = (d[i] - a[i] \* Q[i - 1]) / (b[i] + a[i] \* P[i - 1])  
 P[k] = 0  
 Q[k] = (d[k] - a[k] \* Q[k - 1]) / (b[k] + a[k] \* P[k - 1])  
  
 x = np.zeros(s)  
 x[k] = Q[k]  
  
 **for** i **in** range(s - 2, -1, -1):  
 x[i] = P[i] \* x[i + 1] + Q[i]  
  
 **return** x  
  
**def** variable\_directions(x, y, hx, hy, K, tau, t):  
 U = np.zeros((K, len(x), len(y)))  
 sig\_x = tau/ (2 \* hx\*\*2)  
 sig\_y = tau / (2 \* hy \*\* 2)  
  
 **for** x\_id **in** range(len(x)):  
 **for** y\_id **in** range(len(y)):  
 U[0, x\_id, y\_id] = psi(x[x\_id], y[y\_id])  
  
 **for** k **in** range(K):  
 **for** i **in** range(len(x)):  
 U[k, i, 0] = phi\_2(x[i], t)  
 U[k, i, -1] = phi\_3( x[i], t)  
 **for** k **in** range(K):  
 **for** j **in** range(len(y)):  
 U[k, 0, j] = phi\_0(y[j], t)  
 U[k, -1, j] = phi\_1(y[j], t)  
  
 **for** k **in** range(1, K):  
 U\_temp = np.zeros((len(x), len(y)))  
 a = np.zeros(len(y))  
 b = np.zeros(len(y))  
 c = np.zeros(len(y))  
 d = np.zeros(len(y))  
 t += tau / 2  
 **for** i **in** range(1, len(x) - 1):  
 **for** j **in** range(1, len(y) - 1):  
 a[j] = -sig\_x  
 b[j] = 1 + 2 \* sig\_x  
 c[j] = -sig\_x  
 d[j] = sig\_y \* (U[k - 1, i, j + 1] - 2 \* U[k - 1, i, j] + U[k - 1, i, j - 1]) + U[k - 1, i, j]  
 b[0] = 1  
 c[0] = 0  
 d[0] = phi\_2(x[i], t)  
 a[-1] = 0  
 b[-1] = 1  
 d[-1] = phi\_3( x[i], t)  
 u\_new = run\_through(a, b, c, d, len(d))  
 U\_temp[i] = u\_new  
 **for** j **in** range(len(y)):  
 U\_temp[0, j] = phi\_0(y[j], t)  
 U\_temp[-1, j] = phi\_1(y[j], t)  
  
 a = np.zeros(len(x))  
 b = np.zeros(len(x))  
 c = np.zeros(len(x))  
 d = np.zeros(len(x))  
 t += tau / 2  
  
 **for** j **in** range(1, len(y) - 1):  
 **for** i **in** range(1, len(x) - 1):  
 a[i] = -sig\_y  
 b[i] = 1 + 2 \* sig\_y  
 c[i] = -sig\_y  
 d[i] =sig\_x \* (U\_temp[i + 1, j] - 2 \* U\_temp[i, j] + U\_temp[i - 1, j]) + U\_temp[i, j]  
 b[0] = 1  
 c[0] = 0  
 d[0] = phi\_0( y[j], t)  
 a[-1] = 0  
 b[-1] = 1  
 d[-1] = phi\_1(y[j], t)  
 u\_new = run\_through(a, b, c, d, len(d))  
 **for** i **in** range(len(u\_new)):  
 U[k, i, j] = u\_new[i]  
 **for** i **in** range(len(x)):  
 U[k, i, 0] = phi\_2(x[i], t)  
 U[k, i, -1] = phi\_3(x[i], t)  
  
 **return** U.transpose()  
  
**def** fractional\_step(x, y, hx, hy, K, tau, t):  
 U = np.zeros((K, len(x), len(y)))  
  
 sig\_x = tau / ( hx \*\* 2)  
 sig\_y = tau / ( hy \*\* 2)  
  
 **for** x\_id **in** range(len(x)):  
 **for** y\_id **in** range(len(y)):  
 U[0, x\_id, y\_id] = psi(x[x\_id], y[y\_id])  
  
 **for** k **in** range(K):  
 **for** i **in** range(len(x)):  
 U[k, i, 0] = phi\_2(x[i], t)  
 U[k, i, -1] = phi\_3(x[i], t)  
 **for** k **in** range(K):  
 **for** j **in** range(len(y)):  
 U[k, 0, j] = phi\_0(y[j], t)  
 U[k, -1, j] = phi\_1(y[j], t)  
  
 **for** k **in** range(1, K):  
 U\_temp = np.zeros((len(x), len(y)))  
 a = np.zeros(len(y))  
 b = np.zeros(len(y))  
 c = np.zeros(len(y))  
 d = np.zeros(len(y))  
 t += tau / 2  
 **for** i **in** range(1, len(x) - 1):  
 **for** j **in** range(1, len(y) - 1):  
 a[j] = -sig\_x  
 b[j] = 1 + 2 \* sig\_x  
 c[j] = -sig\_x  
 d[j] = U[k - 1, i, j]  
 b[0] = 1  
 c[0] = 0  
 d[0] = phi\_2(x[i], t)  
 a[-1] = 0  
 b[-1] = 1  
 d[-1] = phi\_3(x[i], t)  
 u\_new = run\_through(a, b, c, d, len(d))  
 U\_temp[i] = u\_new  
 **for** j **in** range(len(y)):  
 U\_temp[0, j] = phi\_0(y[j], t)  
 U\_temp[-1, j] = phi\_1(y[j], t)  
  
 a = np.zeros(len(x))  
 b = np.zeros(len(x))  
 c = np.zeros(len(x))  
 d = np.zeros(len(x))  
 t += tau / 2  
 **for** j **in** range(1, len(y) - 1):  
 **for** i **in** range(1, len(x) - 1):  
 a[i] = -sig\_y  
 b[i] = 1 + 2 \* sig\_y  
 c[i] = -sig\_y  
 d[i] = U\_temp[i, j]  
 b[0] = 1  
 c[0] = 0  
 d[0] = phi\_0(y[j], t)  
 a[-1] = 0  
 b[-1] = 1  
 d[-1] = phi\_1(y[j], t)  
 u\_new = run\_through(a, b, c, d, len(d))  
 **for** i **in** range(len(u\_new)):  
 U[k, i, j] = u\_new[i]  
 **for** i **in** range(len(x)):  
 U[k, i, 0] = phi\_2(x[i], t)  
 U[k, i, -1] = phi\_3(x[i], t)  
  
 **return** U.transpose()  
  
Nx = 20  
Ny = 20  
K = 1000  
time = 1  
hx = (np.pi - 0) / Nx  
hy = (np.pi - 0) / Ny  
x = np.arange(0, np.pi + hx / 2 - 1e-4, hx)  
y = np.arange(0, np.pi + hy / 2 - 1e-4, hy)  
tau = time / K  
T = np.arange(0, time, tau)  
  
t1 = 0  
t = 100  
  
X, Y = np.meshgrid(x, y)  
U\_analytic = analyt\_func( X, Y, T[t])  
U1 = variable\_directions(x, y, hx, hy, K, tau, t1)  
U2 = fractional\_step(x, y, hx, hy, K, tau, t1)  
fig = plt.figure()  
ax = fig.add\_subplot(projection=**'3d'**)  
ax.set\_title(**"Метод переменных направлений"**)  
ax.plot\_wireframe(X, Y, U\_analytic, color=**"red"**, label=**"Точное решение"**)  
ax.plot\_wireframe(X, Y, U1[:, :, t], color=**"blue"**, label=**"Численное решение"**)  
  
ax.set\_xlabel(**"x"**)  
ax.set\_ylabel(**"y"**)  
ax.set\_zlabel(**"U"**)  
ax.legend()  
  
X, Y = np.meshgrid(x, y)  
U\_analytic = analyt\_func( X, Y, T[t])  
U1 = variable\_directions(x, y, hx, hy, K, tau, t1)  
U2 = fractional\_step(x, y, hx, hy, K, tau, t1)  
fig = plt.figure()  
ax = fig.add\_subplot(projection=**'3d'**)  
ax.set\_title(**"Метод дробных шагов"**)  
ax.plot\_wireframe(X, Y, U\_analytic, color=**"red"**, label=**"Точное решение"**)  
  
ax.plot\_wireframe(X, Y, U2[:, :, t], color=**"yellow"**, label=**"Численное решение"**)  
ax.set\_xlabel(**"x"**)  
ax.set\_ylabel(**"y"**)  
ax.set\_zlabel(**"U"**)  
ax.legend()  
plt.show()  
H = np.zeros(3)  
E1 = np.zeros(3)  
E2 = np.zeros(3)  
  
n = 2.5  
**for** i **in** range(3):  
 n = int(n \* 2)  
 hx = (np.pi - 0) / n  
 hy = (np.pi - 0) / n  
 x = np.arange(0, np.pi + hx / 2 - 1e-4, hx)  
 y = np.arange(0, np.pi + hy / 2 - 1e-4, hy)  
 X, Y = np.meshgrid(x, y)  
 U\_analytic = analyt\_func(X, Y, T[t])  
 U1 = variable\_directions(x, y, hx, hy, K, tau, t1)  
 U2 = fractional\_step(x, y, hx, hy, K, tau, t1)  
 H[i] = hx  
 error\_x1 = []  
 error\_x2 = []  
  
  
 **for** j **in** range(len(x)):  
 error\_x1.append(max(abs(U\_analytic[:, j] - U1[:, j,t])))  
 error\_x2.append(max(abs(U\_analytic[:, j] - U2[:, j,t])))  
  
 E1[i] = max(error\_x1)  
 E2[i] = max(error\_x2)  
**from** scipy.interpolate **import** PchipInterpolator  
  
Y1\_reverse = E1[::-1]  
Y2\_reverse = E2[::-1]  
  
X\_reverse = H[::-1]  
  
  
  
pchip\_reverse1 = PchipInterpolator(X\_reverse, Y1\_reverse)  
pchip\_reverse2 = PchipInterpolator(X\_reverse, Y2\_reverse)  
  
  
xnew\_reverse = np.linspace(min(X\_reverse), max(X\_reverse), 1000)  
ynew\_reverse1 = pchip\_reverse1(xnew\_reverse)  
ynew\_reverse2 = pchip\_reverse2(xnew\_reverse)  
  
  
plt.figure(figsize=(10, 6))  
plt.scatter(X\_reverse, Y1\_reverse, label=**''**, zorder=6,c = **'b'**)  
plt.plot(xnew\_reverse, ynew\_reverse1, label=**''**, linewidth=2,c = **'b'**)  
plt.title(**'График огрешности метода переменных направлений'**)  
plt.xlabel(**'h'**)  
plt.ylabel(**'error'**)  
plt.grid(**True**)  
  
plt.figure(figsize=(10, 6))  
plt.scatter(X\_reverse, Y2\_reverse, label=**''**, zorder=6,c = **'b'**)  
plt.plot(xnew\_reverse, ynew\_reverse2, label=**''**, linewidth=2,c = **'b'**)  
plt.title(**'График погрешности метода дробных шагов'**)  
plt.xlabel(**'h'**)  
plt.ylabel(**'error'**)  
plt.grid(**True**)  
  
  
  
plt.show()

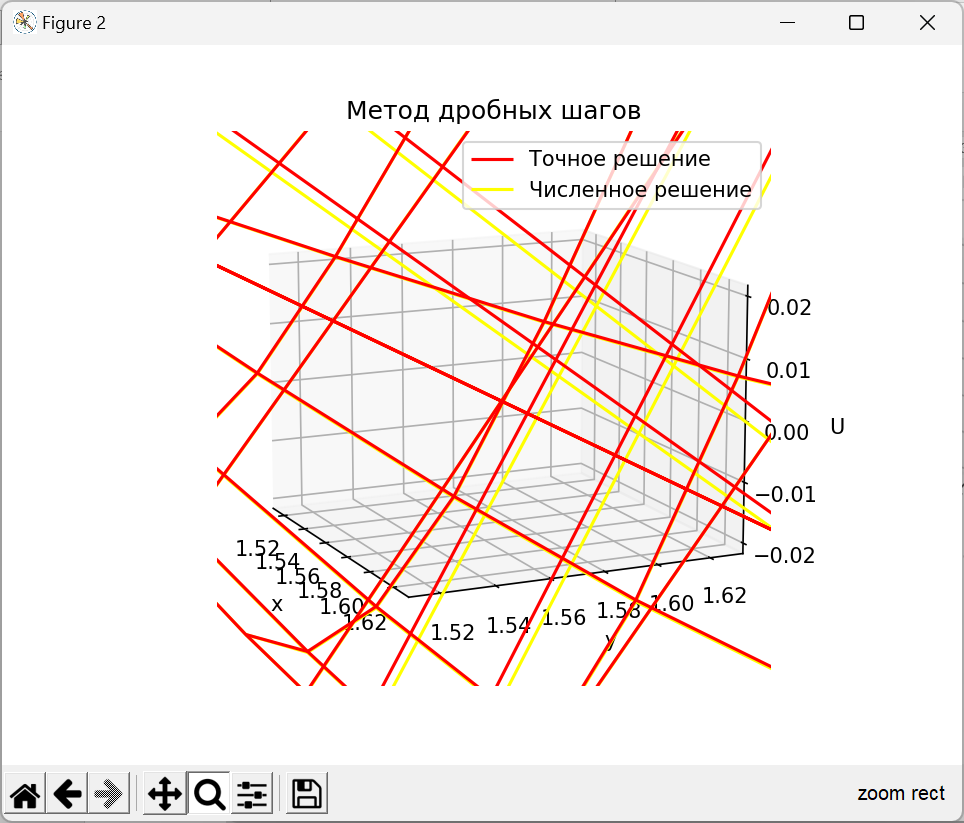
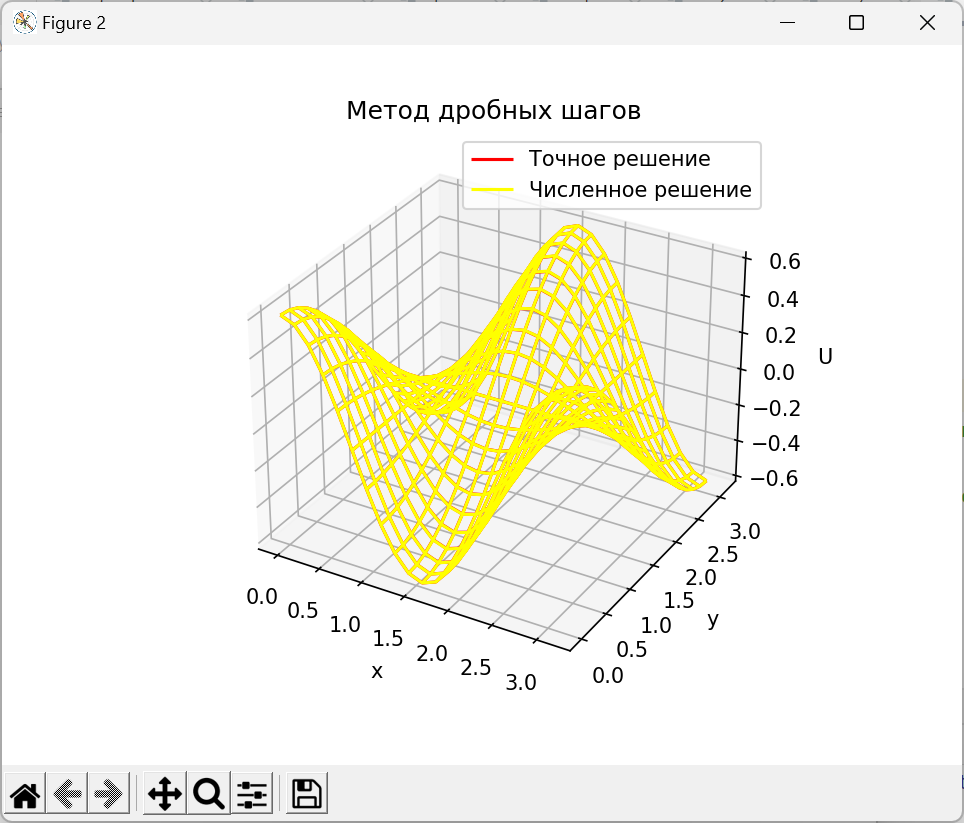
### **Вывод программы**

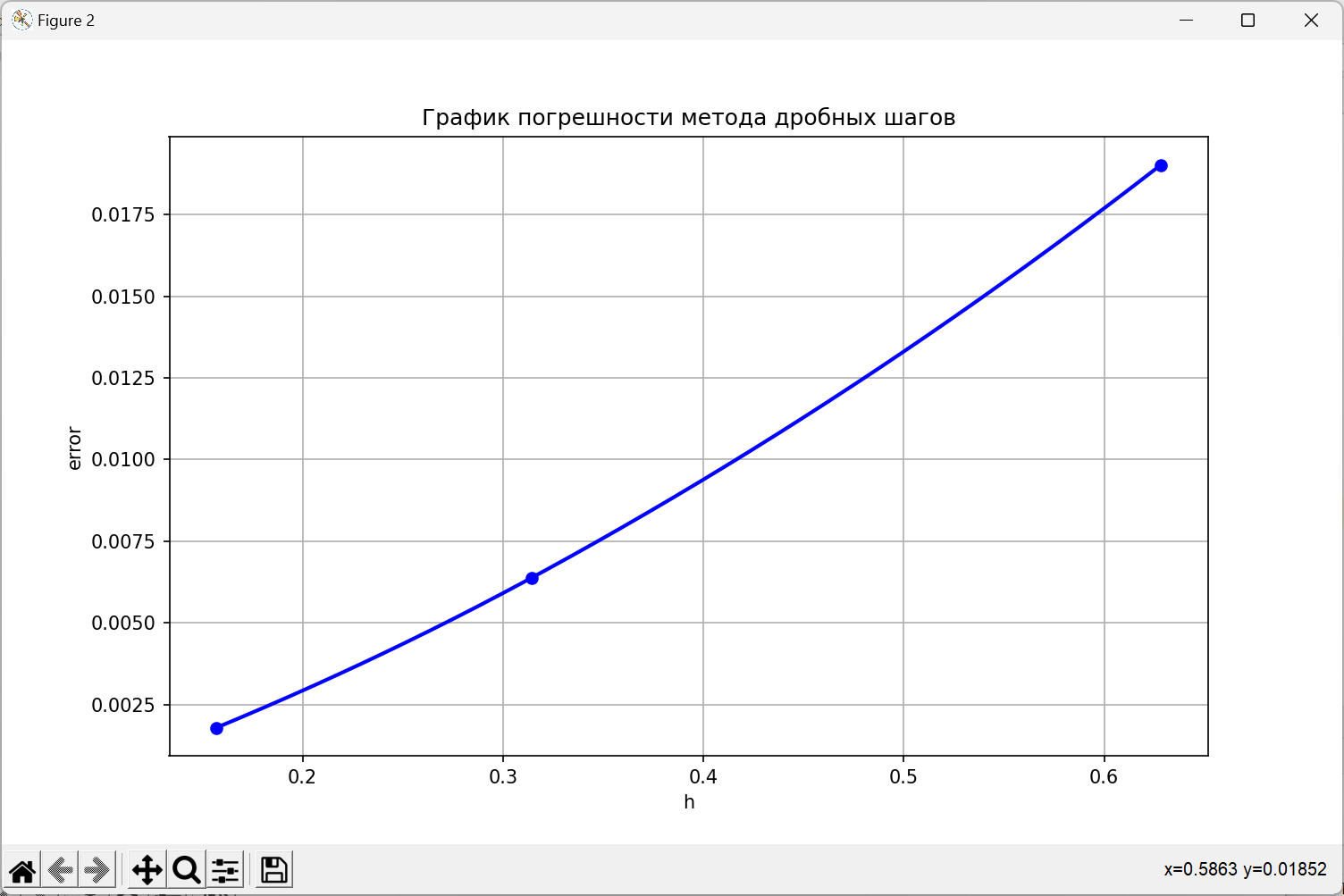
**Метод переменных направлений**





**Метод дробных шагов**





**Заключение:**

Из результатов выполненной работы можно заключить, что для поставленной задачи метод переменных направлений оказался точнее метода дробных шагов. Также было показано, что с увеличением шага дробления закономерно растет погрешность каждого метода.