|  |  |
| --- | --- |
|  | |
| 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 17 | |
|  | |
|  | Выполнил: Черных С. Д. |
|  | Группа: М8О-405Б-20 |
|  | Проверил: доц. Иванов И. Э. |
|  | Дата: |
|  | Оценка: |
|  | |
| Москва, 2023 | |

# **Лабораторная работа №7**

Используя центрально-разностную схему и метод Либмана, решить краевую задачу для дифференциального уравнения эллиптического типа. Вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением U(x,y). Исследовать зависимость погрешности от сеточного параметра h. Реализовать ускорение сходимости итерационной процедуры методом Зейделя и методом верхней релаксации.

|  |
| --- |
| Аналитическое решение: |

### 

### **Теоретические сведения**

На прямоугольнике наложим сетку

. Согласно граничным условиям, мы можем сразу явно заполнить верхнюю и нижнюю строки и первый столбец сетки, так как они задаются граничными условиями первого рода. На этой сетке аппроксимируем дифференциальную задачу во внутренних узлах с помощью отношения конечных разностей:

Выражаем из этого соотношения интересующий нас член, который мы можем найти одним из трех представленных ниже методов:

Метод Либмана:

Метод Зейделя:

Метод простых итераций с верхней релаксацией:

В общем случае метод основан на следующей схеме:

В данной работе:

**Код программы**

Программа выполнена на языке python, в среде разработки Jupyter Notebook. Код программы будет в виде незапущенного кода в Jupyter Notebook, вывод программы, сокращён до графиков, построенных в среде Jupyter Notebook**.**

|  |
| --- |
| Решить краевую задачу для дифференциального уравнения эллиптического типа. Аппроксимацию уравнения произвести с использованием центрально-разностной схемы. Для решения дискретного аналога применить следующие методы: метод простых итераций (метод Либмана), метод Зейделя, метод простых итераций с верхней релаксацией. Вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением . Исследовать зависимость погрешности от сеточных параметров , .  $ u\_x(0, y) = y \ u\_x(\frac{\pi}{2}, y) = 0 \ u(x, 0) = 0 \ u(x, 1) = sin(x) $  import math import typing from typing import List import matplotlib.pyplot as plt import copy import plotly.graph\_objects as go Входные условия x\_left = 0 x\_right = math.pi/2 y\_left = 0 y\_right = 1 a = 1 b = 1 c = 1 d = 0 alpha\_1, betta\_1 = 1, 0 alpha\_2, betta\_2 = 1, 0 alpha\_3, betta\_3 = 0, 1 alpha\_4, betta\_4 = 0, 1  def phi\_1(y:float) -> float:  return y   def phi\_2(y:float) -> float:  return 0  def phi\_3(x:float) -> float:  return 0  def phi\_4(x:float) -> float:  return math.sin(x)  def f(x:float, y:float) -> float:  return 0  def U(x:float, y:float) -> float:  return y \* math.sin(x)  def real\_U(X:list, Y:list) -> list:  n = len(X)  m = len(Y)  U\_true = [[0] \* n for \_ in range(m)]  for k in range(m):  for j in range(n):  U\_true[k][j] = U(X[j], Y[k])  return U\_true Вспомогательные функиции графики  def plot\_graphs(new\_X:list, new\_T:list, found\_U:list, U\_true:list, s:str='') -> None:  plt.plot(new\_X, U\_true[len(new\_T) // 4 ], label='Аналитическое решение')  plt.plot(new\_X, found\_U[len(new\_T) // 4 ], label=s, linestyle='dashdot')  plt.plot(new\_X, U\_true[len(new\_T) // 2 ], label='Аналитическое решение')  plt.plot(new\_X, found\_U[len(new\_T) // 2 ], label=s, linestyle='dashdot')  plt.plot(new\_X, U\_true[len(new\_T) - 1], label='Аналитическое решение')  plt.plot(new\_X, found\_U[len(new\_T) - 1],label=s, linestyle='dashdot')  plt.legend()  график ошибки от h  def local\_error (U\_my:list, U\_true:list) -> float:  return sum([(a - b) \*\* 2 for a, b in zip(U\_my, U\_true)]) / len(U\_my)  def get\_error\_array\_with\_h(N:list, x\_left:float, x\_right:float, y\_left:float, y\_right:float, a:float, b:float, c:float, d:float,   find\_u:typing.Callable, eps:float=0.1, omega:float=-10) -> (list, list): # H, error     H\_x = [x\_right/(n - 1) for n in N]  H\_y = [y\_right/(n - 1) for n in N]  ERROR = []  for n in N:  if omega == -10:  XX, YY, UU = find\_u(x\_left, x\_right, y\_left, y\_right, a, b, c, d, eps=eps, n\_x=n, n\_y=n)  else:  XX, YY, UU = find\_u(x\_left, x\_right, y\_left, y\_right, a, b, c, d=0, eps=eps, n\_x=n, n\_y=n, omega=omega)  U\_true = real\_U(XX, YY)  t = len(YY) // 2  ERROR.append(local\_error(UU[t], U\_true[t]))    return H\_x, H\_y ,ERROR  def h\_error\_plot(H:list, ERROR:list, s:str=' x') -> None:  plt.plot(H, ERROR)  plt.xlabel("h" + s)  plt.ylabel("error")  plt.show()  def frange(start:float, stop:float, step:float) -> float:  while start < stop:  yield start  start += step  def get\_y(y0:float, y\_end:float, h:float) -> list:  return [i for i in frange(y0, y\_end+h, h)]  def get\_x(x\_0:float, x\_l:float, h:float) -> list:  return [i for i in frange(x\_0, x\_l+h, h)]  def interpol(U\_values:List[List[float]], X:List[float], Y:List[float], y\_right:float) -> None:  for i in range(1, len(Y) - 1):  for j in range(1, len(X) - 1):  alpha = Y[i] / y\_right  U\_values[i][j] = phi\_1(X[j]) \* (1 - alpha) + phi\_2(X[j]) \* alpha   def border\_conds(U\_values:List[List[float]], X:List[float], Y:List[float]) -> None:  for i in range(len(Y)):  U\_values[i][0] = phi\_2(Y[i])  U\_values[i][1] = phi\_2(Y[i]) \* (X[1] - X[0])  U\_values[i][-1] = phi\_1(Y[i])  for i in range(len(X)):  U\_values[0][i] = phi\_3(X[i])  U\_values[-1][i] = phi\_4(X[i])  def error(U\_prev:List[List[float]], U\_values:List[List[float]]) -> float:  error\_score = 0  for i in range(len(U\_values)):  for j in range(len(U\_values[0])):  error\_score = max(error\_score, abs(U\_prev[i][j] - U\_values[i][j]))  return error\_score  Отношение конечной разности по схеме  мой случай Метод Лимбмана def limbman(x\_left:float, x\_right:float, y\_left:float, y\_right:float, a:float, b:float, c:float, d:float=0, eps:float=0.1, n\_x:float=10, n\_y:float=10) -> (List[float], List[float], List[List[float]]):  hx = x\_right / (n\_x - 1)  hy = y\_right / (n\_y - 1)  X = get\_x(x\_left, x\_right, hx)  Y = get\_y(y\_left, y\_right, hy)   U\_values = [[0.0] \* len(X) for \_ in range(len(Y))]  U\_prev = [[10.0] \* len(X) for \_ in range(len(Y))]    border\_conds(U\_values, X, Y)  interpol(U\_values, X, Y, y\_right)   cnt = 0  while (error(U\_prev, U\_values) > eps):  U\_prev = copy.deepcopy(U\_values)  for i in range(1, len(Y) - 1):  for j in range(1, len(X) - 1):  U\_values[i][j] = ((U\_prev[i-1][j] + U\_prev[i+1][j]) / hx\*\*2 + (U\_prev[i][j-1] + U\_prev[i][j+1]) / hy\*\*2 + c \* U\_prev[i][j] - f(X[j], Y[i])) \* ((hx\*\*2 \* hy\*\*2) / (2 \* (hx\*\*2 + hy\*\*2)))   cnt +=1   print('Iterations', cnt)  return X, Y, U\_values  X, Y, UU = limbman(x\_left, x\_right, y\_left, y\_right, a, b, c, d=0, eps=0.0005, n\_x=10, n\_y=10) U\_true = real\_U(X, Y)  Моя поверхность  fig = go.Figure(data=[go.Surface(z=UU, x=X, y=Y)]) fig.show()  Точное решение  fig = go.Figure(data=[go.Surface(z=U\_true, x=X, y=Y)]) fig.show()  Две поверхности вместе  fig = go.Figure(data=[go.Surface(z=UU, x=X, y=Y, colorscale='Reds', name='Метод Лимбмана'), go.Surface(z=U\_true, x=X, y=Y, colorscale='Blues', name='Аналитическое решение')]) fig.update\_layout(title='Метод Лимбмана - красный, Аналитическое решение - синий', showlegend=True) fig.show()  Вычисление ошибки и построение графиков  N = [20, 30, 40, 50]  H\_X1, H\_Y1, ERROR\_1 = get\_error\_array\_with\_h(N, x\_left, x\_right, y\_left, y\_right, a, b, c, d=0, eps=0.0005, find\_u=limbman) h\_error\_plot(H\_X1, ERROR\_1) h\_error\_plot(H\_Y1, ERROR\_1, s=' y') Метод Зейделя def Seidel(x\_left:float, x\_right:float, y\_left:float, y\_right:float, a:float, b:float, c:float, d:float=0, eps:float=0.1, n\_x:float=10, n\_y:float=10) -> (List[float], List[float], List[List[float]]):  hx = x\_right / (n\_x - 1)  hy = y\_right / (n\_y - 1)  X = get\_x(x\_left, x\_right, hx)  Y = get\_y(y\_left, y\_right, hy)   U\_values = [[0.0] \* len(X) for \_ in range(len(Y))]  U\_prev = [[10.0] \* len(X) for \_ in range(len(Y))]   border\_conds(U\_values, X, Y)  interpol(U\_values, X, Y, y\_right)     cnt = 0  while (error(U\_prev, U\_values) > eps):  U\_prev = copy.deepcopy(U\_values)  for i in range(1, len(Y) - 1):  for j in range(1, len(X) - 1):  U\_values[i][j] = ((U\_values[i-1][j] + U\_values[i+1][j]) / hx\*\*2 + (U\_prev[i][j-1] + U\_values[i][j+1]) / hy\*\*2 + c \* U\_prev[i][j] - f(X[j], Y[i])) \* ((hx\*\*2 \* hy\*\*2) / (2 \* (hx\*\*2 + hy\*\*2)))   cnt += 1  print('Iterations', cnt)  return X, Y, U\_values  X2, Y2, UU2 = Seidel(x\_left, x\_right, y\_left, y\_right, a, b, c, d=0, eps=0.0005, n\_x=10, n\_y=10) U\_true2 = real\_U(X2, Y2)  Моя поверхность  fig = go.Figure(data=[go.Surface(z=UU2, x=X2, y=Y2)]) fig.show()  Точное решение  fig = go.Figure(data=[go.Surface(z=U\_true2, x=X2, y=Y2)]) fig.show()  Две поверхности вместе  fig = go.Figure(data=[go.Surface(z=UU2, x=X2, y=Y2, colorscale='Reds', name='Метод Зейделя'), go.Surface(z=U\_true2, x=X2, y=Y2, colorscale='Blues', name='Аналитическое решение')]) fig.update\_layout(title='Метод Зейделя - красный, Аналитическое решение - синий', showlegend=True) fig.show()  Вычисление ошибки и построение графиков  N = [20, 30, 40, 50]  H\_X2, H\_Y2, ERROR\_2 = get\_error\_array\_with\_h(N, x\_left, x\_right, y\_left, y\_right, a, b, c, d=0, eps=0.0005, find\_u=Seidel) h\_error\_plot(H\_X2, ERROR\_2) h\_error\_plot(H\_Y2, ERROR\_2, s=' y') Метод простых итераций с верхней релаксацией $$ u^{k+1} = \alpha u^k + \beta \\ u^{k+1} = \omega (\alpha u^k + \beta) + (1 - \omega) u^k \\ u^{k+1}\_{i, j} = \omega \left(\frac{u^{k+1}\_{i-1, j} + u^k\_{i+1, j}}{h\_1^2} + \frac{u^{k+1}\_{i, j-1} + u^k\_{i, j+1}}{h\_2^2} + с u^k\_{i, j} + f\_{i, j}\right) \left(\frac{2}{h\_1^2} + \frac{2}{h\_2^2}\right)^{-1} + (1 - \omega) u^k\_{i, j} $$  def relax(x\_left:float, x\_right:float, y\_left:float, y\_right:float, a:float, b:float, c:float, d:float=0, eps:float=0.1, n\_x:float=10, n\_y:float=10, omega:float=1.5) -> (List[float], List[float], List[List[float]]):  if omega > 2 or omega < 1:  print('Омега введена некорректно, она лежит в интервале (1, 2)')   hx = x\_right / (n\_x - 1)  hy = y\_right / (n\_y - 1)  X = get\_x(x\_left, x\_right, hx)  Y = get\_y(y\_left, y\_right, hy)     U\_values = [[0] \* len(X) for \_ in range(len(Y))]  U\_prev = [[10] \* len(X) for \_ in range(len(Y))]   border\_conds(U\_values, X, Y)  interpol(U\_values, X, Y, y\_right)    cnt = 0  while (error(U\_prev, U\_values) > eps):  U\_prev = copy.deepcopy(U\_values)  for i in range(1, len(Y) - 1):  for j in range(1, len(X) - 1):  U\_values[i][j] = ((U\_values[i-1][j] + U\_prev[i+1][j]) / hx\*\*2 + (U\_values[i][j-1] + U\_prev[i][j+1]) / hy\*\*2 + c \* U\_prev[i][j] - f(X[j], Y[i])) \* ((hx\*\*2 \* hy\*\*2) / (2 \* (hx\*\*2 + hy\*\*2)))  U\_values[i][j] = omega \* U\_values[i][j] + (1 - omega) \* U\_prev[i][j]   cnt += 1  print('Iterations', cnt)  return X, Y, U\_values  X3, Y3, UU3 = relax(x\_left, x\_right, y\_left, y\_right, a, b, c, d=0, eps=0.00005, n\_x=10, n\_y=10) U\_true3 = real\_U(X3, Y3)  Моя поверхность  fig = go.Figure(data=[go.Surface(z=UU3, x=X3, y=Y3)]) fig.show()  Точное решение  Две поверхности вместе  fig = go.Figure(data=[go.Surface(z=U\_true3, x=X3, y=Y3)]) fig.show()  fig = go.Figure(data=[go.Surface(z=UU3, x=X3, y=Y3, colorscale='Reds', name='Метод релаксации'), go.Surface(z=U\_true3, x=X3, y=Y3, colorscale='Blues', name='Аналитическое решение')]) fig.update\_layout(title='Метод релаксации - красный, Аналитическое решение - синий', showlegend=True) fig.show()  Вычисление ошибки и построение графиков  N = [20, 30, 40, 50]  H\_X3, H\_Y3, ERROR\_3 = get\_error\_array\_with\_h(N, x\_left, x\_right, y\_left, y\_right, a, b, c, d=0, eps=0.0005, find\_u=relax, omega=1.2) h\_error\_plot(H\_X3, ERROR\_3) h\_error\_plot(H\_Y3, ERROR\_3, s=' y') |

### **Вывод программы**

Для вывода программы был взят вывод графиков, чтобы не дублировать код.

|  |
| --- |
| Метод Лимбмана, график численного решения  **C:\Users\jrtit\Desktop\Numer2\newplot.png**график аналитического решения  **C:\Users\jrtit\Desktop\Numer2\newplot.png**  Две поверхности на одном графике  **C:\Users\jrtit\Desktop\Numer2\newplot.png**  **C:\Users\jrtit\Desktop\Numer2\newplot1.png**График ошибки в зависимости от шага(метод Лимбмана)    Метод Зейделя, график численного решения и график аналитического решения    Две поверхности на одном графике    График ошибки в зависимости от шага(метод Зейделя)    Метод верхней релаксации, график численного решения и график аналитического решения      Две поверхности на одном графике      График ошибки в зависимости от шага(метод верхней релаксации) |

**Количество итераций и сетки**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Количество итераций | | |
| Nx | Ny | Метод Либмана | Метод Зейделя | Метод простых итераций с верхней релаксацией |
| 20 | 20 | 158 | 140 | 72 |
| 30 | 30 | 272 | 246 | 134 |
| 40 | 40 | 375 | 344 | 200 |
| 50 | 50 | 464 | 430 | 267 |

**Заключение:**

В данной лабораторной работе построены разностно-итерационные методы (Либмана, Зейделя, метод простых итераций с верхней релаксацией) для дифференциальных уравнений эллиптического типа с двумя пространственными переменными.Схемы были программно реализованы на языке программирования Python в среде разработки Jupyter Notebook.

Аналитическое решение представляет собой плоскость, все методы сходятся к нему, достигая поставленной точности и завершая процесс.

Но методы отличаются количеством итераций: за наименьшее количество итераций выполняется метод простых итераций с верхней релаксацией, далее метод Зейделя, затем метод простых итераций.