|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  |
| МИНОБРНАУКИ РОССИИ | | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | |
| **Институт кибернетики** | | | |
| **Кафедра высшей математики** | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **КУРСОВАЯ РАБОТА** | | | | | | | |
| **по дисциплине** | | | | | | | |
| *«* | *Сеточные модели уравнений с частными производными* | | | | | | *»* |
|  | | | | | | | |
| **Тема курсовой работы** | | | **«** | **Решение задачи Коши для системы** | | | |
| **дифференциальных уравнений, построение многочлена наилучшего приближения.** | | | | | | | **»** |
|  | | | | | | | |
| Студент группы | | **КМБО-03-16** | | |  | Антонов Артем Евгеньевич | |
|  | |  | | |  |  | |
|  | |  | | |  | к.т.н., доцент | |
| Руководитель курсовой работы | | | | |  | **Сенявин Михаил Маркович** | |
|  | | | | |  |  | |

|  |  |  |
| --- | --- | --- |
| Работа представлена к защите | «\_\_»\_\_\_\_\_\_\_201\_\_\_ г. |  |
| *(подпись студента)* |
|  |  |  |
| «Допущен к защите» | «\_\_»\_\_\_\_\_\_\_201\_\_\_ г. |  |
| *(подпись руководителя)* |

Москва 2019

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | | |  | | |
| МИНОБРНАУКИ РОССИИ | | | | | | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | | | | | |
| **Институт кибернетики** | | | | | | | |
| **Кафедра высшей математики** | | | | | | | |
|  | | | | **Утверждаю** | | | |
|  | | | | Заведующий  кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_*Ю.И.Худак* | | | |
|  | | | | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_2019 г. | | | |
| **ЗАДАНИЕ** | | | | | | | |
| **на выполнение курсовой работы** | | | | | | | |
| **по** **дисциплине** «*Сеточные модели уравнений с частными производными*» | | | | | | | |
|  | | | | | | | |
| Студент Антонов А. Е. Группа *КМБО-03-16* | | | | | | | |
|  | | | | | | | |
| 1. **Тема: «Решение задачи Коши для системы дифференциальных уравнений, построение многочлена наилучшего приближения** | | | | | | | |
| 1. **Исходные данные:**   Решить задачу Коши для системы:  По таблице значений функции построить многочлен второй степени наилучшего приближения по критерию наименьших квадратов. | | | | | | | |
|  | | | | | | | |
| 1. **Срок представления к защите курсовой работы:** **до** « » \_\_\_\_\_\_\_\_\_\_\_ 2019 г. | | | | | | | |
|  | | | | | | | |
| Задание на курсовую  работу выдал | | | «\_\_\_»\_\_\_\_\_\_2019 г. | | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* | | *(\_\_\_\_\_\_\_\_\_\_\_\_)* |
| Задание на курсовую  работу получил | | | «\_\_\_»\_\_\_\_\_\_2019 г. | | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* | | *(\_\_\_\_\_\_\_\_\_\_\_\_)* |

**Содержание**

Задание на курсовую работу . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4 Теоретические сведения. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

Решение . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

Список литературы . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .15 Приложение. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

**Задание на курсовую работу**

Решить задачу Коши для системы:

По таблице значений функции построить многочлен второй степени наилучшего приближения по критерию наименьших квадратов.

**Теоретические сведения**

**Метод Рунге-Кутты**

Рассмотрим задачу Коши для системы обыкновенных дифференциальных уравнений первого порядка.

Рассмотрим Метод Рунге — Кутты четвёртого порядка с постоянным шагом интегрирования.

Приближённое значение в последующих точках вычисляется по итерационной формуле:

где {\displaystyle h}h — величина шага сетки по {\displaystyle x}x.

Этот метод имеет четвёртый порядок точности. Это значит, что ошибка на одном шаге имеет порядок {\displaystyle O(h^{5})}, а суммарная ошибка на конечном интервале интегрирования имеет порядок{\displaystyle O(h^{4})} {\displaystyle O(h^{5})}.

### **Оценка точности численного решения ОДУ по правилу Рунге**:

Формула .

Метод Рунге-Кутты четвёртого порядка, поэтому формула имеет следующий вид:

Отличительной чертой правила Рунге является то, что с его помощью можно оценить не глобальную, а локальные ошибки дискретизации, то есть погрешность метода на каждом отдельном шаге. Абсолютную погрешность можно принять за максимум модуля локальных ошибок.

**Формулы метода Рунге-Кутты 4-ого порядка для систем двух уравнений**

Сведём систему к одномерному случаю, используя векторную запись.

**Метод наименьших квадратов**

Пусть задана таблично в узлах функция . При этом значения функции определены с некоторой погрешностью. Пусть также из физических соображений известен вид функции, которой должны приближенно удовлетворять табличные точки, например, многочлен степени n:

У которого неизвестны коэффициенты . Будем их находить из условия минимум квадратичного отклонения многочлена от таблично заданной функции:

Необходимые условия экстремума имеют вид:

Эту систему для удобства преобразуют к виду:

Которая называется нормальной системой метода наименьших квадратов и представляет собой систему линейных алгебраических уравнений относительно коэффициентов . Решив её, построим многочлен , приближающий таблично заданную функцию и минимизирующий квадратичное отклонение.

**Решение**

Поставленная задача решается в два этапа:

* 1. Решение системы дифференциальных уравнений.
  2. Построение многочлена второй степени наилучшего приближения по критерию наименьших квадратов по таблице значений функции

Рассмотрим подробнее каждый из пунктов.

**Этап 1: Решение системы дифференциальных уравнений**

Будем искать решение с помощью метода Рунге-Кутты четвёртого порядка для системы ОДУ

Реализация метода на языке Python.

**def** Runge\_Kutta(arr\_x, h):

arr\_y\_1 = np.zeros(len(arr\_x))

arr\_y\_1[0] = 0.2

arr\_y\_2 = np.zeros(len(arr\_x))

arr\_y\_2[0] = 0

**for** i **in** range(1, len(arr\_x)):

k1 = f\_1(arr\_x[i-1], arr\_y\_2[i-1])

q1 = f\_2(arr\_x[i-1], arr\_y\_1[i-1])

k2 = f\_1(arr\_x[i-1] + h/2, arr\_y\_2[i-1] + k1 \* h / 2)

q2 = f\_2(arr\_x[i-1] + h/2, arr\_y\_1[i-1] + q1 \* h / 2)

k3 = f\_1(arr\_x[i-1] + h/2, arr\_y\_2[i-1] + k2 \* h / 2)

q3 = f\_2(arr\_x[i-1] + h/2, arr\_y\_1[i-1] + q2 \* h / 2)

k4 = f\_1(arr\_x[i-1] + h, arr\_y\_2[i-1] + k3 \* h)

q4 = f\_2(arr\_x[i-1] + h, arr\_y\_1[i-1] + q3 \* h)

arr\_y\_1[i] = arr\_y\_1[i-1] + h\*(k1 + 2\*k2 + 2\*k3 + k4)/6

arr\_y\_2[i] = arr\_y\_2[i-1] + h\*(q1 + 2\*q2 + 2\*q3 + q4)/6

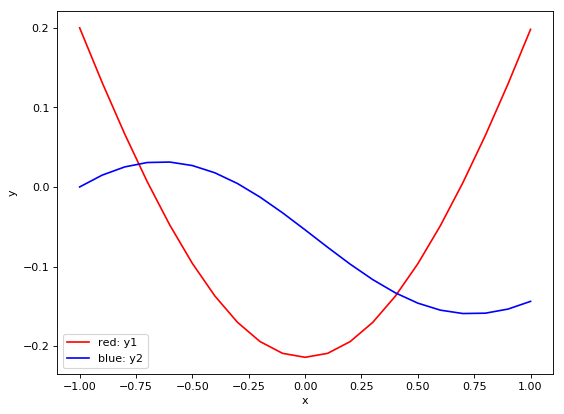
**return** (arr\_y\_1, arr\_y\_2)

Результат работы программы занесём в таблицу

Значений функций в таблице округлены до пятого знака после запятой.

|  |  |  |
| --- | --- | --- |
| x |  |  |
| -1.0 | 0.2 | 0.0 |
| -0.9 | 0.13117810283646342 | 0.01486641310587516 |
| -0.8 | 0.06645320307263429 | 0.02519488000265758 |
| -0.7 | 0.006490191162412555 | 0.030720870968703844 |
| -0.6 | -0.04797008923529361 | 0.03128847179618074 |
| -0.5 | -0.09612204831901494 | 0.026900489253026603 |
| -0.4 | -0.13712005921512233 | 0.01776468727671313 |
| -0.3 | -0.17012103210487384 | 0.004324796841897722 |
| -0.2 | -0.19434652728009552 | -0.012733889496557975 |
| -0.1 | -0.20915980091983924 | -0.03250897020097512 |
| -0.0 | -0.21414432228739486 | -0.05393254079635151 |
| 0.1 | -0.2091633178589974 | -0.07584143243060668 |
| 0.2 | -0.19438088948150417 | -0.09705588529707272 |
| 0.3 | -0.1702366109843797 | -0.11645754173777412 |
| 0.4 | -0.13738193788445566 | -0.13305895977827104 |
| 0.5 | -0.09659856856920326 | -0.1460573942958158 |
| 0.6 | -0.04872013378954925 | -0.15486659912804943 |
| 0.7 | 0.0054287990498801675 | -0.15912347318796408 |
| 0.8 | 0.06507210284327786 | -0.1586713195247461 |
| 0.9 | 0.12950249637271136 | -0.1535261959770652 |
| 1.0 | 0.19808847828278048 | -0.1438351226510872 |

Выведем две функции на одном графике:



**Этап 2: Построение многочлена второй степени наилучшего приближения по критерию наименьших квадратов по таблице значений функции**

Нормальная система метода наименьших квадратов в случае квадратичного многочлена в матричном виде имеет вид:

Решим её методом Зейделя.

**def** norma\_matr(arr):

sums = [sum(abs(arr[i])) **for** i **in** range(len(arr))]

**return** max(sums)

**def** norma\_vect(v):

**return** sum([abs(el) **for** el **in** v])

**def** Seidel(A, b):

n = len(A)

B = np.zeros((n, n))

d = np.zeros(n)

**for** i **in** range(n):

**for** j **in** range(n):

**if** i == j:

B[i][i] = 0

**else**:

B[i][j] = -A[i][j] / A[i][i]

d[i] = b[i]/A[i][i]

eps = 0.000001

x\_new = np.copy(d)

**for** i **in** range(len(x\_new)):

x\_new[i] = np.dot(B[i], x\_new) + d[i]

x = np.copy(d)

**while** norma\_vect(np.dot(A, x\_new) - b) > eps:

x = np.copy(x\_new)

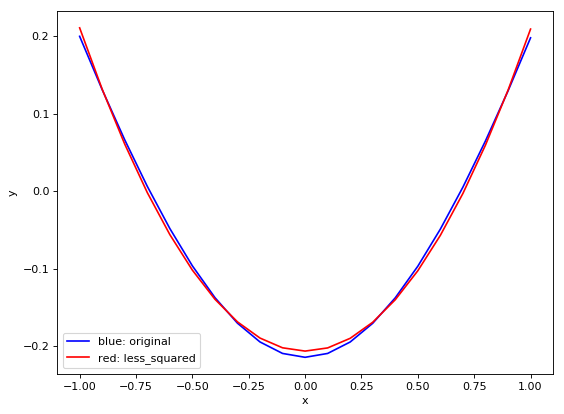
**for** i **in** range(len(x\_new)):

x\_new[i] = np.dot(B[i], x\_new) + d[i]

**return** x\_new

Для получения решения с точностью потребовалось 32 итерации.

Искомый многочлен:

****

На рисунке синим выделен график функции, полученный в ходе решения системы дифференциальных уравнений, а красным — график приближающего многочлена второй степени.

**Оценка погрешности**

Для оценки погрешности было применено правило Рунге, информация о котором находится в разделе «Теоретические сведения». Для оценки погрешности потребуется таблица функций с шагом 0.2

|  |  |  |
| --- | --- | --- |
| x |  |  |
| -1.0 | 0.2 | 0.0 |
| -0.8 | 0.06664010443849772 | 0.03162826280703116 |
| -0.6 | -0.04772425617668502 | 0.04346930862796779 |
| -0.4 | -0.13685834771590255 | 0.034137682226079755 |
| -0.2 | -0.19408003413073172 | 0.00560398473474082 |
| 0 | -0.21388359622854777 | -0.036022884977272185 |
| 0.2 | -0.19408653625200264 | -0.08166794484779974 |
| 0.4 | -0.1369390352366365 | -0.12168806955212055 |
| 0.6 | -0.04802190166061818 | -0.14823746683492872 |
| 0.8 | 0.06604200617595049 | -0.1567782732409422 |
| 1.0 | 0.1992633904809114 | -0.14620567560153397 |

Применим правило Рунге и получим таблицу

|  |  |  |
| --- | --- | --- |
| x | Правило Рунге для | Правило Рунге для |
| -1.0 | 0.0 | 0.0 |
| -0.8 | 1.246009105756174e-05 | 0.0004288921869582389 |
| -0.6 | 1.6388870573906072e-05 | 0.0008120557887858033 |
| -0.4 | 1.7447433281318295e-05 | 0.0010915329966244417 |
| -0.2 | 1.776620995758704e-05 | 0.001222524948753253 |
| 0 | 1.7381737256472186e-05 | 0.001193977054605288 |
| 0.2 | 1.9623548633435074e-05 | 0.0010258626966181989 |
| 0.4 | 2.952684318794363e-05 | 0.0007580593484100321 |
| 0.6 | 4.654880859540494e-05 | 0.0004419421528747136 |
| 0.8 | 6.466022217817558e-05 | 0.00012620308558692708 |
| 1.0 | 7.832747987539524e-05 | 0.00015803686336311832 |

Опираясь на все вычисленные значения, можно сделать вывод, что

Вычислим сумму квадратов отклонений:

Учтём, что табличные данные, по которым строился аппроксимирующий многочлен, были найдены с погрешностью.

То есть , где

Тогда

Подставив все значения, получим:

Таким образом, погрешность вычисления коэффициентов аппроксимирующего многочлена, ровно, как и всей задачи, можно оценить как .

**Список использованной литературы**

1. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. – 4-е изд. – М.:БИНОМ. Лаборатория знаний, 2006. – 636с
2. **Галилеев М.М., Гончар Л.И., Грузина Т.Н.** Численные методы: учеб. пособие.- СПб.: СПбГИЭУ, 2012. 125 с
3. Гидаспов В. Ю., Иванов И. Э., Ревизников Д. Л. Численные методы: сборник задач. – М.:Дрофа., 2007. 99с

**Приложение**

**import** numpy **as** np

**import** scipy.sparse

**import** scipy.linalg

**import** matplotlib.pyplot **as** plt

**def** norma\_matr(arr):

sums = [sum(abs(arr[i])) **for** i **in** range(len(arr))]

**return** max(sums)

**def** norma\_vect(v):

**return** sum([abs(el) **for** el **in** v])

**def** Seidel(A, b):

n = len(A)

B = np.zeros((n, n))

d = np.zeros(n)

**for** i **in** range(n):

**for** j **in** range(n):

**if** i == j:

B[i][i] = 0

**else**:

B[i][j] = -A[i][j] / A[i][i]

d[i] = b[i]/A[i][i]

eps = 10\*\*(-8)

x\_new = np.copy(d)

**for** i **in** range(len(x\_new)):

x\_new[i] = np.dot(B[i], x\_new) + d[i]

x = np.copy(d)

**while** norma\_vect(np.dot(A, x\_new) - b) > eps:

x = np.copy(x\_new)

**for** i **in** range(len(x\_new)):

x\_new[i] = np.dot(B[i], x\_new) + d[i]

**return** x\_new

**def** Runge\_Kutta(arr\_x, h):

arr\_y\_1 = np.zeros(len(arr\_x))

arr\_y\_1[0] = 0.2

arr\_y\_2 = np.zeros(len(arr\_x))

arr\_y\_2[0] = 0

**for** i **in** range(1, len(arr\_x)):

k1 = f\_1(arr\_x[i-1], arr\_y\_2[i-1])

q1 = f\_2(arr\_x[i-1], arr\_y\_1[i-1])

k2 = f\_1(arr\_x[i-1] + h/2, arr\_y\_2[i-1] + k1 \* h / 2)

q2 = f\_2(arr\_x[i-1] + h/2, arr\_y\_1[i-1] + q1 \* h / 2)

k3 = f\_1(arr\_x[i-1] + h/2, arr\_y\_2[i-1] + k2 \* h / 2)

q3 = f\_2(arr\_x[i-1] + h/2, arr\_y\_1[i-1] + q2 \* h / 2)

k4 = f\_1(arr\_x[i-1] + h, arr\_y\_2[i-1] + k3 \* h)

q4 = f\_2(arr\_x[i-1] + h, arr\_y\_1[i-1] + q3 \* h)

arr\_y\_1[i] = arr\_y\_1[i-1] + h\*(k1 + 2\*k2 + 2\*k3 + k4)/6

arr\_y\_2[i] = arr\_y\_2[i-1] + h\*(q1 + 2\*q2 + 2\*q3 + q4)/6

**return** (arr\_y\_1, arr\_y\_2)

f\_1 = **lambda** x, y\_2: x/np.sqrt(1 + x\*x + y\_2\*y\_2)

f\_2 = **lambda** x, y\_1: y\_1/np.sqrt(1 + x\*x + y\_1\*y\_1)

h = 0.1

xx = np.arange(-1, 1+h, h)

yy\_1, yy\_2 = Runge\_Kutta(xx, h)

plt.figure(num=None, figsize=(8, 6), dpi=80, facecolor='w', edgecolor='k')

plt.plot(xx, yy\_1, 'r', xx, yy\_2, 'b')

plt.xlabel('x')

plt.ylabel('y')

plt.legend(("red: y1", "blue: y2"))

M = np.zeros((3, 3))

**for** i **in** range(3):

**for** j **in** range(3):

M[i][j] = np.sum(xx \*\* (i+j))

Y = np.array([np.sum(yy\_1), np.sum(yy\_1 \* xx), np.sum(yy\_1 \* xx\*\*2)])

**print**(np.linalg.solve(M, Y))

*#coef = np.linalg.solve(M, Y)*

coef = Seidel(M, Y)

yy\_1\_ls = coef[0] + coef[1]\*xx + coef[2]\*xx\*\*2

plt.figure(num=None, figsize=(8, 6), dpi=80, facecolor='w', edgecolor='k')

plt.plot(xx, yy\_1, 'b', xx, yy\_1\_ls, 'r')

plt.xlabel('x')

plt.ylabel('y')

plt.legend(("blue: original", "red: less\_squared"))

*# оценка точности по правилу Рунге*

h2 = 0.2

xx\_h2 = np.arange(-1, 1+h2, h2)

yy\_1\_h2, yy\_2\_h2 = Runge\_Kutta(xx\_h2, h2)

difference\_1 = [abs(yy\_1[2\*i]-yy\_1\_h2[i])/15 **for** i **in** range(len(yy\_1\_h2))]

difference\_2 = [abs(yy\_2[2\*i]-yy\_2\_h2[i])/15 **for** i **in** range(len(yy\_2\_h2))]

**print**(max(difference\_2))

*# вычислим сумму квадратов отклонений*

**print**(sum((yy\_1 - yy\_1\_ls)\*\*2))