Отчет по лабораторной работе №5

Снопов П.М.

16 мая 2020 г.

3 курс, 3 группа

Лабораторная работа №5

Метод универсальной дифференциальной прогонки для линейных уравнений второго порядка.

Вариант 9

1. Постановка задачи Метод универсальной дифференциальной прогонки для линейных уравнений второго порядка:
Имеется линейное дифференциальное уравнение второго порядка

$$y''(x) + p(x)y'(x) = q(x)y(x) + f(x),\ x \in [a, b]$$

С краевыми условиями:

$$\alpha_0 y(a) + \beta_0 y'(a) = A$$
$$\alpha_1 y(b) + \beta_1 y'(b) = B$$

Используя метод дифференциальной прогонки, решить данное дифференциальное уравнение

2. Метод решения Задача решается в 2 этапа:

Прямая прогонка Предполагая, что $\beta_0 \neq 0$, решаем 2 независимые друг от друга задачи Коши на [a,b]:

$$\begin{cases} z_1' = -z_1^2 - p(x)z_1 + q(x) \\ z_1(a) = -\dfrac{\alpha_0}{\beta_0} \end{cases} \begin{cases} z_2' = z_2[z_1 + p(x)] + f(x) \\ z_2(b) = \dfrac{A}{\beta_0} \end{cases}$$

Если $\beta_0 = 0$, то $\alpha_0 \neq 0$Если, конечно$A \neq 0$, и тогда решаем 2 другие независимые друг от друга задачи Коши на [a,b]:

$$\begin{cases} z_1' = -z_1^2 q(x) + z_1 p(x) + 1 \\ z_1(a) = -\dfrac{\beta_0}{\alpha_0} \end{cases} \begin{cases} z_2' = -z_1[z_2 q(x) + f(x)] \\ z_2(b) = -\dfrac{A}{\alpha_0} \end{cases}$$

Обратная прогонка При обратной прогонке получаем приближенное решение исходной краевой задачи. Если $\beta_0 \neq 0$, то решаем следующую задачу Коши:

$$\begin{cases} y' = z_1 y + z_2 \\ y(b) = \dfrac{B - \beta_1 z_2(b)}{\alpha_1 + \beta_1 z_1(b)} \end{cases}$$

Если же $\alpha_0 \neq 0$, то задача Коши имеет вид:

$$\begin{cases} y' = \dfrac{y - z_2}{z_1} \\ y(b) = \dfrac{B z_1(b) + \beta_1 z_2(b)}{\beta_1 + \alpha_1 z_1(b)} \end{cases}$$

3. Основные процедуры Основные функции, используемые при решении задачи:

```
def RungeKutta(f: Callable, h: float, x: float, y=0) -> float:
```

Функция, соответствующая методу Рунге-Кутта 4 порядка(формула Кутта-Менсона)

```
def forward(X: dict, b_cond1: list, interval: list) -> dict,dict:
```

Функция, реализующая прямую прогонку

```
def backward(X: dict, b_cond1: list, b_cond2: list, interval: list, Z1
    : dict, Z2: dict) -> dict:
```

Функция, реализующая обратную прогонку

```
def write_data(X:dict, Y:dict, Yprime:dict):
```

Функция записывающая полученные данные

```
def solve():
```

Основная функция, осуществляющая решение задачи.