Отчёт по задаче №2.44

Орлов Даниил Ильич

December 2024

1 Постановка задачи

Требуется решить численно следующее дифференциальное уравнение:

$$\begin{cases} \dfrac{d^2x}{dx^2} + x^2 + \lambda t^2 = 1, \\ 0 < t < 1, \\ t = 0 : x = 0, \dot{x} = 1; \\ t = 1 : x = 0. \end{cases} \tag{1}$$

И найти минимальное λ, такое, при котором существует нетривиальное решение

2 Теоретическая часть

1. Будем решать данную задачу методом Рунге-Кутта с 5-м порядком погрешности. Для этого мы возьмём наше уравнвение и начальные условия, чтобы получить задачу Коши.

2. Приведём наше уравнение к нужному виду, введя эти обозначения: $x = x_1$ и $\dot{x} = x_2$. Получим следующее:

$$\begin{cases} \dot{x_2} = 1 - \lambda t^2 - x_1^2, \\ \dot{x_1} = x_2, \\ 0 < t < 1, \\ t = 0 : x_1 = 0, x_2 = 1, \\ t = 1 : x_1 = 0; \end{cases}$$

3. Для нахождения λ мы запустим наш метод, начиная с λ_0, проводя итерации по $\Delta\lambda$ с конечным значением в λ_{end}.

4. Чтобы получить нужные лямбда, мы построим график зависимости решения задачи в точке $x(1)$ от λ и нас будет интересовать точки λ, близкие к нулю

5. Вопрос минимальности, на мой взгляд не совсем уместен. Учитывая, что мы получаем малые значения, мы будем близки к точке $x(1) = 0$, но так как у нас лишь дискретное количество точек, то при уменьшении шага итерации $\Delta\lambda$ мы получим точки, более точные(в плане минимальности), чем получили до этого

3 Практическая часть

1. Ранее мы уже упомянали, что решать будем методом Рунге-Кутты 5-го порядка. Ниже написаны вложенные формулы Рунге, необходимые для решения задачи:
   - $\vec{k}_1 = hf(t, \vec{x})$
   - $\vec{k}_2 = hf(t + \frac{h}{2}, \vec{x} + \frac{\vec{k}_1}{2})$
   - $\vec{k}_3 = hf(t + \frac{h}{2}, \vec{x} + \frac{(\vec{k}_1+\vec{k}_2)}{4})$
   - $\vec{k}_4 = hf(t + h, \vec{x} - \vec{k}_2 + 2\vec{k}_3)$
   - $\vec{k}_5 = hf(t + \frac{2h}{3}, \vec{x} + \frac{(7\vec{k}_1+10\vec{k}_2+\vec{k}_4)}{27})$
   - $\vec{k}_6 = hf(t + \frac{h}{5}, \vec{x} + \frac{(28\vec{k}_1-125\vec{k}_2+546\vec{k}_3+54\vec{k}_4-378\vec{k}_5)}{625})$
2. Ошибку будем вычислять следующим образом: $\vec{r} = -\frac{(42\vec{k}_1+224\vec{k}_3+21\vec{k}_4-162\vec{k}_5-125\vec{k}_6)}{336}$
3. Ошибка на шаге: $||\vec{r}||$
4. Вычисление значения функции на шаге: $\vec{x}(t+h) = \vec{x}(t) + \frac{(\vec{k}_1+4\vec{k}_3+\vec{k}_4)}{6}$
5. Тестирование программы. Возьмем два примера из книги Э.Хайрер, С.Нёрсетт и Г.Ваннера "Решение обыкновенных дифференциальных уравнений Нежёсткие задачи ". А именно:

$$\begin{cases} \dot{x}_1 = x_2 x_3, & x_1(0) = 0, \\ \dot{x}_2 = -x_1 x_3, & x_2(0) = 1, \\ \dot{x}_3 = -0.51 x_1 x_2, & x_3(0) = 1, \\ 0 \leqslant t \leqslant 20. \end{cases}$$

   Решая систему, можем получить следующий график:

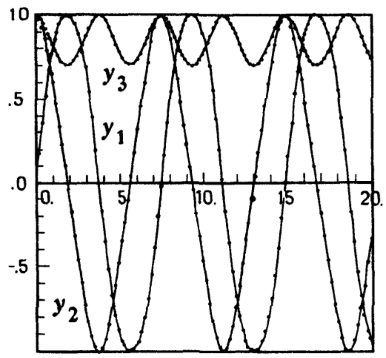

Рис. 1: Из книги.

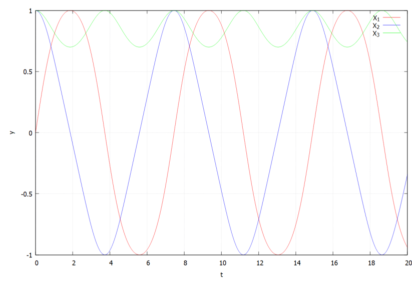

Рис. 2: Мой вариант при решении задачи.

Также рассмотрим ещё одну задачу из этой же книги:

$$\begin{cases} \dot{x}_1 = x_2, & x_1(0) = 2, \\ \dot{x}_2 = (1 - x_1^2)x_2 - x_1, & x_2(0) = 0, \\ 0 \leqslant t \leqslant 20. \end{cases}$$

Её решения приведём следующим графиком:

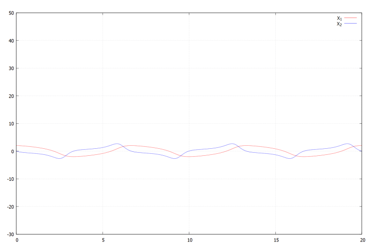

Рис. 3: Общий вид.

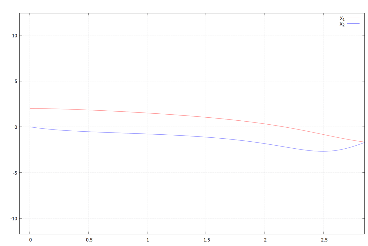

Рис. 4: Приближение.

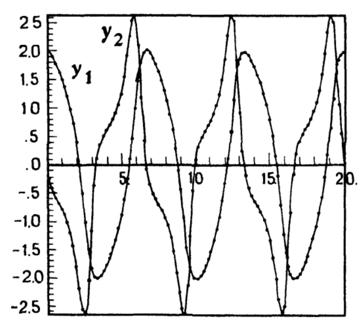

Рис. 5: Из книги.

Графики, которые я привёл, очень схожи, можно сделать вывод, что моя программа выдаёт корректный результат. Можно взять также другие готовые методы рунге, и просто сравнивать ошибку в результатах при вычислении решения уравнения, но я считаю, что результат будет схожим.

4 Результаты:

Программа работает довольно эффективно. Она гибкая, чтобы метод вычислял решение функции разной размерности, нужно лишь задать саму функцию, и размерность. Также учитывая, что я в решении использовал язык C++, а не чистый C(хотя разница в моей программе лишь в форме вывода), программа считает не много медленнее, чем та же программа на C. Причиной этому стал вывод программы на консоль, или запись файла, так как использовалась стандартная библиотека STL(в ней тот же std::cout - замедляет программу из-за того, что это объект ООП, а сам вывод - это перегруженный метод Б «< который тоже замедляет). Но несмотря на все, программа весьма "шустро"решает задачу.

Сначало запускаем программу для $\lambda = -100$ с шагом 10^{-3}, получаю следующие графики:

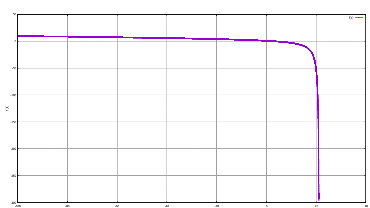

Рис. 6: Общий вид.

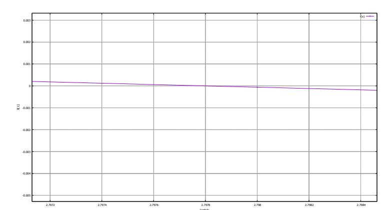

Рис. 7: Приближённый вариант.

Если посмотреть на графики, то одно из ближайших значений(итерация с шагом 10^{-3}) это $\lambda = 2.797$.

Рассмотрим теперь итерирование с шагом 10^{-4}, получаем следующий графики:

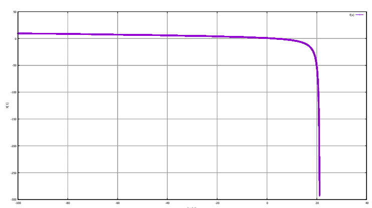

Рис. 8: Общий вид.

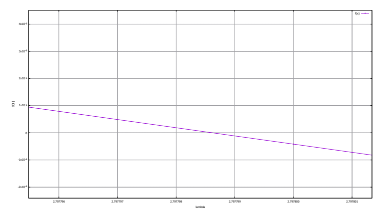

Рис. 9: Приближённый вариант.

Уже в этом случае, мы получаем, что минимальное значение, которое нам подходит, оно является $\lambda = 2.7977$

В заключении также хочу сказать, что на точность, скорость работы программы влияет(существенно влияет) погрешность, которую мы принимаем на шаге, в моей работе ошибку я брал очень жёсткую, а именно $tol = 2 \cdot 10^{-16}$