

Отчёт по 1 заданию практикума на ЭВМ студента 422 группы Сойфера Петра Юрьевича.

Задание 1.1.

а) Найти "машинный эпсилон": $\min \varepsilon > 0$ такой, что $1 + \varepsilon > 1$.

Метод решения: начиная с некоторого числа ε_0 , результат сложения которого с единицей заведомо даст в машинном представлении число большее 1 (в реализации алгоритма было взято $\varepsilon_0 = 1$), итеративно делить его на 2 (так как числа в машинном представлении опираются на двоичную систему счисления) и снова проверять условие $1 + \varepsilon > 1$.

По окончании алгоритма будет получено численное значение "машинного эпсилон", которое сверяется с имеющимся в стандартных библиотеках языка C значением `__DBL_EPSILON__` $= 2^{-52} \approx 2.22 \cdot 10^{-16}$, отвечающим наименьшему возможному двоичному машинному представлению нормализованного числа типа `double`.

Результат работы программы:

```
machine epsilon = 2.22045e-16
in comparison with __DBL_EPSILON__ our epsilon is true
```

б) Найти $\min X$ такой, что $X + 1 = X$.

Метод решения: аналогично предыдущему алгоритму, начиная с некоторого числа X_0 , результат сложения которого с единицей заведомо не даст в машинном представлении это же число (в реализации алгоритма было взято $X_0 = 1$), итеративно умножать его на 2, проверяя условие $X + 1 \neq X$.

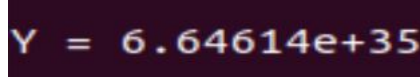
Результат работы программы:

```
X = 4.5036e+15
```

в) Найти $\min Y$ такой, что $Y + 10^{20} = Y$.

Метод решения: почти дословно повторяя алгоритм пункта (б), начиная с $Y_0 = __\text{DBL_EPSILON}___$, итеративно умножать его на 2, проверяя условие $Y + 10^{20} \neq Y$. Для проверки полученного числа разумно предположить, что оно должно быть примерно на 20 порядков больше числа из пункта (б) в силу похожести условий. Гипотеза подтвердилась.

Результат работы программы:



```
Y = 6.64614e+35
```

Задание 1.2.

а) Показать неустойчивость прямого метода рекуррентного вычисления следующего интеграла:

$$I_n = \int_0^1 \frac{x^n}{x+6} dx = \int_0^1 x^{n-1} dx - 6 \int_0^1 \frac{x^{n-1}}{x+6} dx = \frac{1}{n} - 6 I_{n-1}.$$

Метод решения: для доказательства неустойчивости требовалось с помощью этого рекуррентного соотношения алгоритмически найти I_{31} и убедиться, что накопление машинной ошибки при вычислениях даёт ошибку порядка $(-6)^{31}$, что вкупе с реальным численным значением $I_{31} \approx 0+$ не позволяет вычислить этот интеграл таким способом.

Для задания рекуррентного цикла требуется найти I_0 :

$$I_0 = \int_0^1 \frac{dx}{x+6} = \int_0^1 \frac{d(x+6)}{x+6} = \ln(x+6) \Big|_{x=0}^{x=1} = \ln\left(\frac{7}{6}\right)$$

После этого рекуррентным циклом с помощью заданного соотношения проводится 31 итерация, после которой выводится значение найденного таким способом числа I_{31} . Оно, конечно же, не будет в точности равно $(-6)^{31} \approx -1.33 \cdot 10^{24}$, так как в цикле на промежуточных вычислениях к исходному числу добавляется поправка $\frac{1}{n}$, и после очередного умножения эти неточности приобретают довольно значимую величину — конечное значение I_{31} на порядки меньше, чем $(-6)^{31}$, но тем не менее неустойчивость алгоритма очевидна.

Результат работы программы:

```
straight method: I_31 = -5.88448e+07
```

б) Показать устойчивость обратного метода рекуррентного вычисления интеграла из пункта (а):

$$I_{n-1} = \frac{1}{6n} - \frac{I_n}{6}.$$

Метод решения: Для задания рекуррентного цикла в обратном методе требуется найти какой-нибудь I_n . Возьмём $n = 62$, чтобы число итераций для нахождения I_{31} было таким же, как в пункте (а). По аналитическим соображениям (площадь под графиком и поведение функции возле 0, 1 и между ними) $I_n \rightarrow 0$, $n \rightarrow +\infty$. Следовательно, примем, что $I_{62} = 0$ с достаточно хорошим приближением.

Рекуррентный цикл принципиально ничем не отличается от алгоритма, реализованного в пункте (а).

Результат работы программы:

```
reverse method: I_31 = 0.0046290481
```

в) Вычислить интеграл I_{31} как интегральную сумму Дарбу с равномерным разбиением отрезка $[0, 1]$ на 1000 точек:

$$I_{31} = \sum_{k=0}^{N-1} (x_{k+1} - x_k) \cdot f(x_k) = \sum_{k=0}^{999} \frac{1}{1000} \cdot \frac{x_k^{31}}{x_k + 6}.$$

Метод решения: В начале цикла заводятся две переменные — сумма, равная нулю и аргумент $x_0 = 0$. Затем к сумме итеративно прибавляется очередное слагаемое. Интересно, что в виду строгой монотонности функции если в теле цикла сначала добавлять к аргументу 0.001, а затем вычислять сумму, то будет получаться верхняя сумма Дарбу, а если эти строчки кода поменять местами — то нижняя. Таким образом можно получить довольно точную верхнюю и нижнюю оценки искомого интеграла.

Результат работы программы:

```
lower Darboux sum: 0.00441271482012
upper Darboux sum: 0.00455557196298
```

Задание 1.3.

Придумать элементарную функцию $f(x)$, дифференцируемую на некотором отрезке, и для выбранной точки x_0 этого отрезка найти погрешности $R_1 = \left| f'(x_0) - \frac{f(x_0+h)-f(x_0)}{h} \right|$ при разных $h = 1, 0.1, 0.01, \dots, 10^{-20}$.

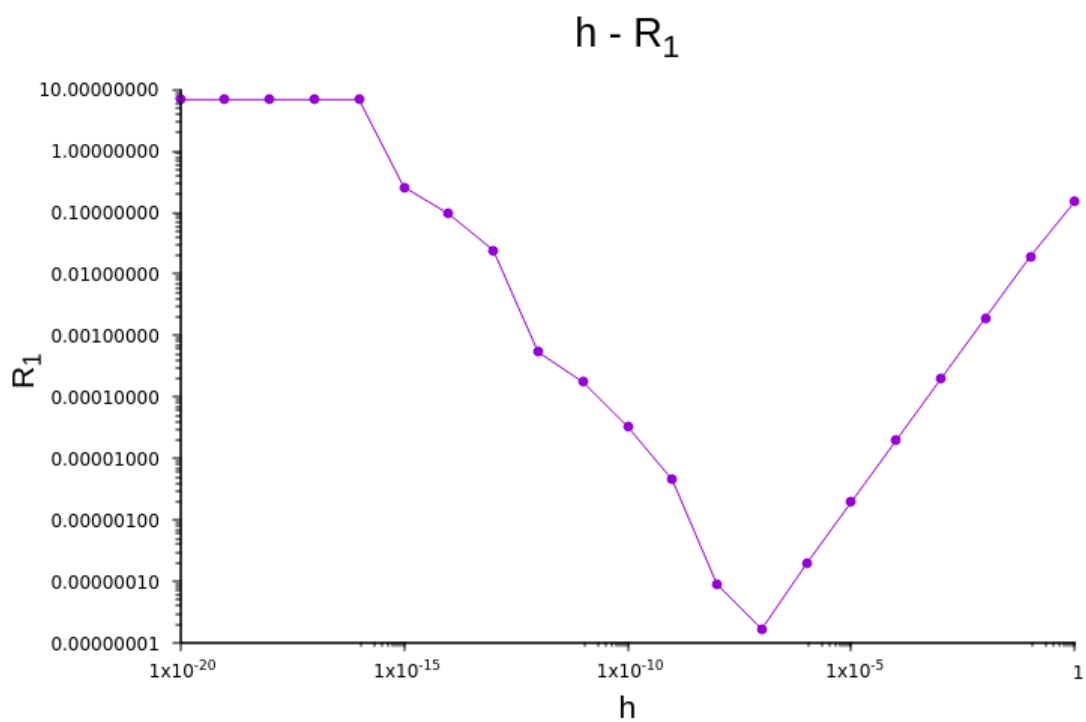
Решение. Функция $f(x)$ и её производная:

$$f(x) = 7\sqrt{x^2 + x + 1} + 9e^{-x} \sin x - \frac{1}{\ln^2 x}$$
$$f'(x) = \frac{7}{2} \cdot \frac{2x + 1}{\sqrt{x^2 + x + 1}} + 9e^{-x} (\cos x - \sin x) + \frac{2}{x \ln^3 x}$$

Точка $x_0 = \pi$; $f'(\pi) \approx 6.845544$. Главная часть алгоритма — итеративный цикл по h , на каждом шаге уменьшающий h в 10 раз от 1 до 10^{-20} и вычисляющий R_1 по заданной формуле. Результаты выводятся в файл "output" без расширения (для более удобного импорта в ЛАТЭХ пришлось добавить расширение ".csv").

Для построения графика зависимости R_1 от h использовался gnuplot. Из-за большого разброса значений R_1 оси взяты в логарифмической шкале. На полученных данных можно наглядно увидеть теоретически выведеную закономерность: при относительно больших h график стремится к линейной функции, а при достаточно малых — гиперболически растёт. Последние 5 значений R_1 совпадают с $f'(x_0)$ — вероятно, это можно объяснить тем, что при столь малых h машинная арифметика уже не различает $f(x_0+h)$ и $f(x_0)$, поэтому $R_1 \approx |f'(x_0)|$. На чуть больших значениях видно нормальное строго монотонное возрастание, что подтверждает совпадение теории и практики.

Результаты работы программы:



h	R_1
1	0.151779
0.1	0.0193125
$1 \cdot 10^{-2}$	0.00194347
$1 \cdot 10^{-3}$	0.000194411
$1 \cdot 10^{-4}$	0.0000194417
$1 \cdot 10^{-5}$	0.00000194395
$1 \cdot 10^{-6}$	0.000000192814
$1 \cdot 10^{-7}$	0.0000000167964
$1 \cdot 10^{-8}$	0.000000089785
$1 \cdot 10^{-9}$	0.00000452874
$1 \cdot 10^{-10}$	0.0000329505
$1 \cdot 10^{-11}$	0.000175059
$1 \cdot 10^{-12}$	0.000535484
$1 \cdot 10^{-13}$	0.0243335
$1 \cdot 10^{-14}$	0.0953878
$1 \cdot 10^{-15}$	0.259884
$1 \cdot 10^{-16}$	6.84554
$1 \cdot 10^{-17}$	6.84554
$1 \cdot 10^{-18}$	6.84554
$1 \cdot 10^{-19}$	6.84554
$1 \cdot 10^{-20}$	6.84554

Задание 1.4.

а) Решить методом Эйлера следующую задачу Коши:

$$\begin{cases} \dot{y} = 2t \\ y_0 = y(t_0) = 0 \end{cases}, \quad t \in [0, 10].$$

Использовать три равномерных разбиения по t : с шагом $h = 0.1, 0.01$ и 0.001 .

Решение. Метод Эйлера (Рунге-Кутты первого порядка) основан на приближённом интегрировании дифференциального уравнения:

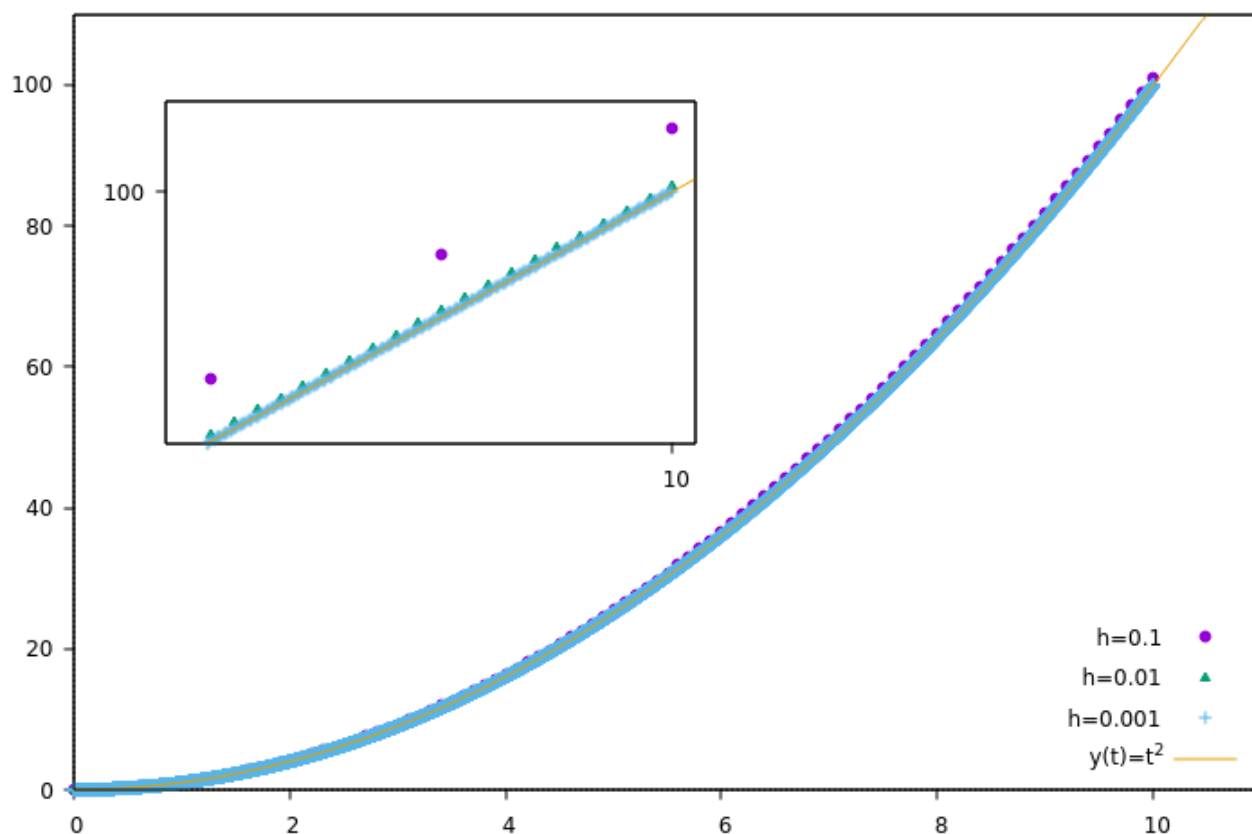
$$\begin{aligned} \int_{t_0}^{t_1} \dot{y}(t) dt &= \int_{t_0}^{t_1} f(t, y(t)) dt \\ y(t_1) &= y(t_0) + f(t_0, y(t_0)) \cdot (t_1 - t_0) \end{aligned}$$

Процесс рекуррентно продолжается с t_0 на t_1 и далее:

$$\begin{aligned} h &= t_{k+1} - t_k, \quad y_k = y(t_k); \\ y_{k+1} &= y_k + f(t_k, y_k) \cdot h \end{aligned}$$

В данном конкретном примере задача Коши позволяет легко найти явный аналитический ответ: $y = t^2$. Рекуррентный цикл реализуется стандартно, аналогично подобным в вышеописанных задачах. На каждой итерации в файл записываются очередные значения t_k и y_k . Для каждого из трёх разбиений создаётся отдельный файл, затем полученные данные визуализируются в gnuplot. Из-за достаточно малых шагов разбиения на основной массе данных отклонения от аналитического решения почти незаметны, поэтому концевой участок графика приближен отдельно.

Результат работы программы:



б) Решить методом Эйлера следующую задачу Коши:

$$\begin{cases} y' = g(x) = f'(x) \\ y_0 = y(x_0) = f(x_0) \end{cases} \quad x \in [x_0, X],$$

где $f(x)$, $f'(x)$ взяты из Задания 1.3, а промежуток $[x_0, X]$ — некоторый выбранный отрезок, на котором функция дифференцируема. Использовать три равномерных разбиения по x : с шагом $h = 0.1, 0.01$ и 0.001 .

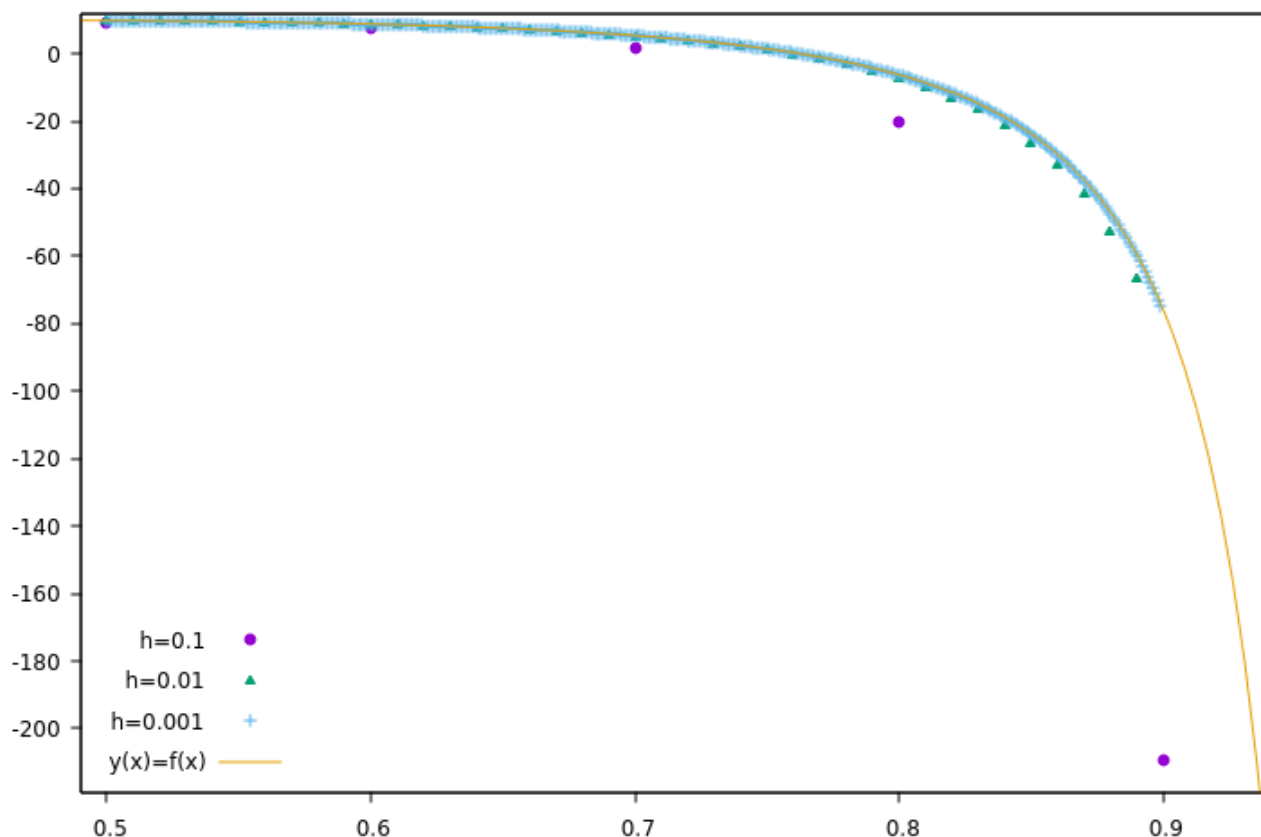
Решение. В точке $x = 1$ функция терпит разрыв (логарифм в знаменателе обращается в нуль), поэтому будем рассматривать отрезок $[0.5, 0.9]$:

$$x_0 = 0.5, \quad X = 0.9, \quad y_0 = f(0.5) \approx 9.79584.$$

Метод Эйлера реализуется так же, как и в предыдущем пункте. Функция решения $f(x)$ теперь более сложная с точки зрения вычислений, поэтому будет хуже приближаться всеми разбиениями — разница в значениях достаточно велика, чтобы увидеть её на конце графика без приближения.

Для разбиения $h = 0.1$ ошибка на конце столь велика, что уже сравнима с значением функции.

Результат работы программы:



Задание 1.5 "Метод Эйлера использовать нельзя".

Решить методом Эйлера следующую задачу Коши для системы дифференциальных уравнений:

$$\begin{cases} \dot{x} = z, & x(0) = 0 \\ \dot{z} = -x, & z(0) = 1 \end{cases}$$

Для доказательства неустойчивости метода:

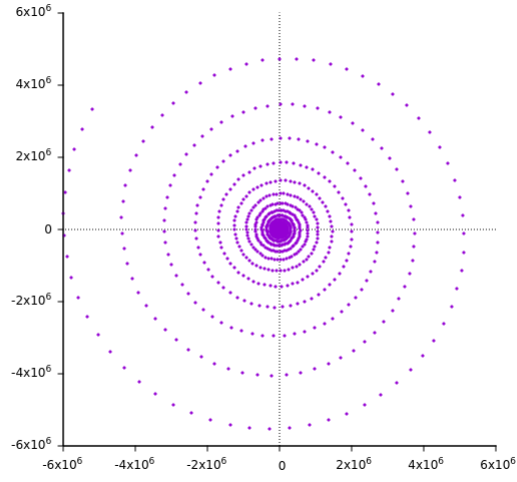
1) получить таблицу с значениями расхождения численного и аналитического решения $|\tilde{x}(T) - x(T)|$, $|\tilde{z}(T) - z(T)|$ для $T = 2\pi, 10\pi, \dots, 100\,000\pi$ и шагов разбиения $h = 0.1, 0.01, 0.001$.

2) построить на плоскости (x, z) 2-3 примера полученных поточечных численных приближений аналитического решения $x = \sin t$, $z = \cos t$.

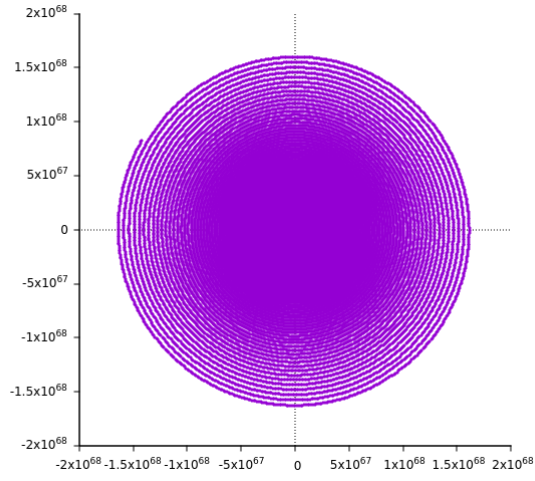
Решение. Метод Эйлера реализовывался аналогично заданию 1.4, с отличием только в размерности системы. Для того, чтобы решение удовлетворяло системе, необходимо последовательно прибавлять слагаемые в рекуррентном соотношении $\vec{y}_{n+1} = \vec{y}_n + h \cdot \vec{f}(t_n, \vec{y}_n)$, то есть на итерации цикла прибавлять сразу и к x_n , и к z_n , что и было реализовано в программе.

Результаты работы программы:

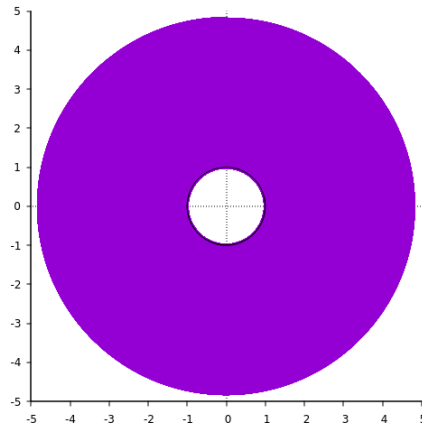
T	h	$ \tilde{x}(T) - x(T) $	$ \tilde{z}(T) - z(T) $
2pi	0.1	$5.55 \cdot 10^{-3}$	0.37
10pi	0.1	$9.73 \cdot 10^{-2}$	3.79
100pi	0.1	$5.18 \cdot 10^6$	$3.32 \cdot 10^6$
1000pi	0.1	$6.29 \cdot 10^{67}$	$4.24 \cdot 10^{67}$
10000pi	0.1	NaN	NaN
100000pi	0.1	NaN	NaN
2pi	0.01	$6.82 \cdot 10^{-3}$	$3.19 \cdot 10^{-2}$
10pi	0.01	$3.54 \cdot 10^{-3}$	0.17
100pi	0.01	$4.68 \cdot 10^{-2}$	3.81
1000pi	0.01	$6.45 \cdot 10^5$	$6.6 \cdot 10^6$
10000pi	0.01	$1.42 \cdot 10^{68}$	$8.26 \cdot 10^{67}$
100000pi	0.01	NaN	NaN
2pi	0.001	$8.15 \cdot 10^{-4}$	$3.15 \cdot 10^{-3}$
10pi	0.001	$6.4 \cdot 10^{-5}$	$1.58 \cdot 10^{-2}$
100pi	0.001	$7.37 \cdot 10^{-4}$	0.17
1000pi	0.001	$3.37 \cdot 10^{-3}$	3.81
10000pi	0.001	66,406.8	$6.64 \cdot 10^6$
100000pi	0.001	$1.69 \cdot 10^{67}$	$1.65 \cdot 10^{68}$



$$h = 0.1, T = 100 \pi.$$



$$h = 0.01, T = 10\,000 \pi.$$



$$h = 0.001, T = 1\,000 \pi.$$

Задание 1.6.

а) Решить задачу Коши из 1.4 (б) методом Рунге-Кутты 5-го порядка с коэффициентами Дормана-Принса. Показать увеличение точности при равномерных разбиениях с шагом $h = 0.1, 0.01, 0.001$. Вычислить погрешность по правилу Рунге.

Решение. Поскольку дифференциальное уравнение из пункта (б) задачи 1.4 тривиально, то есть не содержит y в правой части и по сути является простейшим случаем уравнения в разделяющихся переменных (мы сразу получаем ответ при взятии определённого интеграла), здесь можно считать k_s только с учётом части для аргумента x :

$$k_s = g(x_0 + c_s h),$$

$$\text{где } g(x) = f'(x) = \frac{7}{2} \cdot \frac{2x+1}{\sqrt{x^2+x+1}} + 9 e^{-x} (\cos x - \sin x) + \frac{2}{x \ln^3 x}.$$

Так как функция терпит разрыв в точке $x = 1$, а отрезок интегрирования $I = [0.5, 0.9]$, то, по-видимому, при сравнительно крупном разбиении $h = 0.1$ на конце отрезка функция уже плохо подходит для численного интегрирования (это видно в том числе и по графику метода Эйлера из Задания 1.4 (б)) и потому разность численного и аналитического значений выдаёт неадекватно большое число. Впрочем, при уменьшении шага видно, что ситуация значительно улучшается до приемлемых значений ошибки, что опять-таки согласуется с методом Эйлера.

Правило Рунге вычисления погрешности между аналитическим и численным решением:

$$y_k - \tilde{y}_{k(h,h)} \sim \frac{\tilde{y}_{k(h,h)} - \tilde{y}_{k(2h)}}{1 - 2^s}.$$

Эти значения вычисляются на всей траектории решения дифференциального уравнения. Поскольку в данной модельной задаче точное аналитическое решение нам известно, можно сравнить, насколько "эквивалентны" эти величины в том смысле, в котором это указано в формуле выше.

Вероятно, в силу неудачности выбора отрезка интегрирования (конец рядом с точкой разрыва) и сравнительно крупных разбиений, более-менее достоверную корреляцию значений можно заметить только в последнем случае с $h = 0.001$. На мелких разбиениях ввиду большого количества точек приведены только последние 10 значений.

Результаты работы программы:

h	$ y_k - \widetilde{y}_{k(h,h)} $	$\frac{\widetilde{y}_{k(h,h)} - \widetilde{y}_{k(2h)}}{1-2^s}$
0.1	1.04	$2.66 \cdot 10^{-2}$
0.1	3.38	0.12
0.1	11.63	0.68
0.1	69.55	NaN
0.1	$1.91 \cdot 10^{46}$	NaN
$1 \cdot 10^{-2}$	2.38	0.17
$1 \cdot 10^{-2}$	2.81	0.19
$1 \cdot 10^{-2}$	3.35	0.23
$1 \cdot 10^{-2}$	4.04	0.26
$1 \cdot 10^{-2}$	4.91	0.31
$1 \cdot 10^{-2}$	6.04	0.37
$1 \cdot 10^{-2}$	7.55	0.45
$1 \cdot 10^{-2}$	9.58	0.54
$1 \cdot 10^{-2}$	12.39	0.67
$1 \cdot 10^{-2}$	16.39	0.85
$1 \cdot 10^{-3}$	1.43	0.57
$1 \cdot 10^{-3}$	1.48	0.58
$1 \cdot 10^{-3}$	1.52	0.59
$1 \cdot 10^{-3}$	1.56	0.6
$1 \cdot 10^{-3}$	1.61	0.62
$1 \cdot 10^{-3}$	1.66	0.63
$1 \cdot 10^{-3}$	1.71	0.64
$1 \cdot 10^{-3}$	1.76	0.66
$1 \cdot 10^{-3}$	1.81	0.67
$1 \cdot 10^{-3}$	1.87	0.69

б) Решить задачу Коши для системы ОДУ из Задания 1.5 методом Рунге-Кутты 5-го порядка с коэффициентами Дормана-Принса. Сравнить точность с методом Эйлера для аналогичных значений $T = 2\pi, 10\pi, \dots, 100\,000\pi$ и шагов разбиения $h = 0.1, 0.01, 0.001$.

Решение. В этой задаче, в отличие от пункта (а), система уравнений автономна:

$$\dot{\vec{y}} = \vec{f}(\vec{y})$$

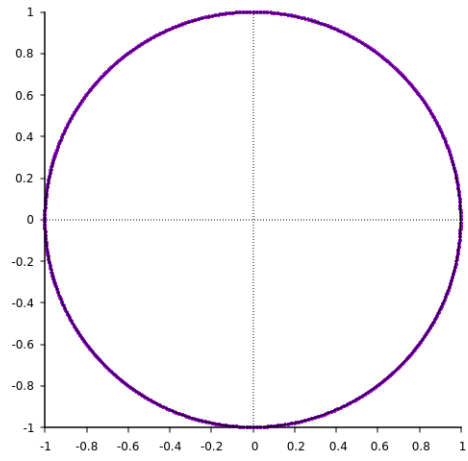
и, следовательно, в формулах для $\vec{k}_s = (k_{sx}, k_{sz})$ будут участвовать только свёртки a_{ij} с $k_{j(x,z)}$ (покоординатно):

$$k_{sx} = f_x(x_0 + h(a_{s1}k_{1x} + \dots + a_{s,s-1}k_{(s-1)x}), z_0 + h(a_{s1}k_{1z} + \dots + a_{s,s-1}k_{(s-1)z}))$$

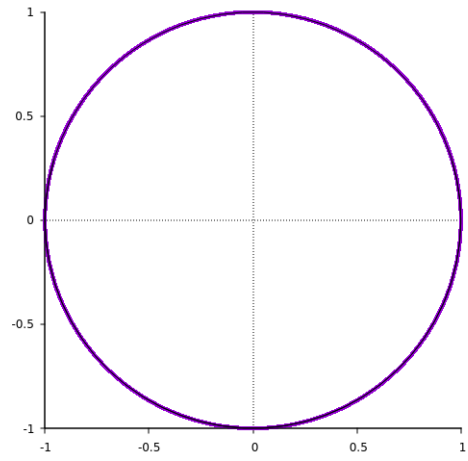
Здесь под x и f_x понимается не аргумент вектор-функции \vec{y} и не частная производная $\frac{\partial f}{\partial x}$, а первая координата этого двумерного вектора и его функционального выражения соответственно. Для второй компоненты k_{sz} выражение аналогичное.

Результаты работы программы:

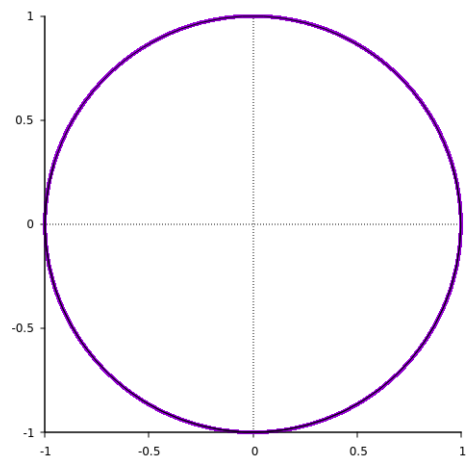
T	h	$ \tilde{x}(T) - x(T) $	$ \tilde{z}(T) - z(T) $
2pi	0.1	$1.68 \cdot 10^{-2}$	$1.41 \cdot 10^{-4}$
2pi	0.01	$6.81 \cdot 10^{-3}$	$2.32 \cdot 10^{-5}$
2pi	0.001	$8.15 \cdot 10^{-4}$	$3.32 \cdot 10^{-7}$
10pi	0.1	$8.4 \cdot 10^{-2}$	$3.53 \cdot 10^{-3}$
10pi	0.01	$4.07 \cdot 10^{-3}$	$8.3 \cdot 10^{-6}$
10pi	0.001	$7.35 \cdot 10^{-5}$	$2.7 \cdot 10^{-9}$
100pi	0.1	$4.07 \cdot 10^{-2}$	$8.3 \cdot 10^{-4}$
100pi	0.01	$7.35 \cdot 10^{-4}$	$2.7 \cdot 10^{-7}$
100pi	0.001	$7.35 \cdot 10^{-4}$	$2.7 \cdot 10^{-7}$
1000pi	0.1	$7.35 \cdot 10^{-3}$	$3.56 \cdot 10^{-5}$
1000pi	0.01	$7.35 \cdot 10^{-3}$	$2.7 \cdot 10^{-5}$
1000pi	0.001	$3.46 \cdot 10^{-4}$	$6 \cdot 10^{-8}$
10000pi	0.1	$7.34 \cdot 10^{-2}$	$2.78 \cdot 10^{-3}$
10000pi	0.01	$3.46 \cdot 10^{-3}$	$6 \cdot 10^{-6}$
10000pi	0.001	$4.64 \cdot 10^{-4}$	$1.08 \cdot 10^{-7}$
100000pi	0.1	$3.48 \cdot 10^{-2}$	$1.47 \cdot 10^{-3}$
100000pi	0.01	$4.64 \cdot 10^{-3}$	$1.08 \cdot 10^{-5}$
100000pi	0.001	$2.64 \cdot 10^{-3}$	$3.49 \cdot 10^{-6}$



$$h = 0.1, T = 100 \pi.$$



$$h = 0.01, T = 10\,000 \pi.$$



$$h = 0.001, T = 1\,000 \pi.$$

Задание 1.7 "Выбор шага".

Решить задачу Коши для системы ОДУ из Задания 1.5 методом Рунге-Кутты 5-го порядка с коэффициентами Дормана-Принса и выбором оптимальной длины шага h . Для заданных значений допустимой погрешности $tol = 10^{-7}, 10^{-9}, 10^{-11}$ и периода $T = 100\pi, \dots, 100\,000\pi$ найти величины $|\tilde{x}(T) - x(T)|$, $|\tilde{z}(T) - z(T)|$, количество шагов N_{steps} и величину шага h . Оценить глобальную погрешность, посчитать числа Рунге R_x, R_z .

Решение. При движении по решению (увеличивая время t) в точках разбиения вычисляются два значения для каждой из координат: x и \hat{x} (соответственно, z и \hat{z}). Для этого используются два разных массива из таблицы Дормана-Принса — b и \hat{b} . Коэффициенты k_i из метода Рунге-Кутты вычисляются аналогично предыдущей задаче. Имея два набора значений, можно находить локальную погрешность err в данной конкретной точке разбиения.

По формуле (4.6) на странице 177 книги "Решение обыкновенных дифференциальных уравнений. Нежёсткие задачи" под авторством Э. Хайре-ра, С. Нёрсетта и Г. Ваннера локальную ошибку предлагается искать так:

$$err = \frac{1}{2^p - 1} \max_{i=1, \dots, n} \frac{|x_i - \hat{x}_i|}{d_i},$$

где d_i — масштабирующий множитель (поскольку здесь изучается не относительная, а абсолютная ошибка, принимается $d_i = 1$), а p — порядок метода (в нашем случае $p = 5$). Авторы отмечают, что допустимо использовать и другие нормы векторов, например, евклидову:

$$err = \sqrt{(x_1 - \hat{x}_1)^2 + \dots + (x_n - \hat{x}_n)^2}.$$

Тесты показывают, что евклидова норма даёт значительно более точные данные численных решений (особенно для $tol = 10^{-7}$), поэтому в реализации использовалась именно эта формула.

Далее величина err сравнивается с tol , и в зависимости от соотношений между ними длина шага либо увеличивается, либо уменьшается, либо не изменяется. Общая формула для вычисления новой оптимальной длины шага h_{new} после предыдущей итерации с шагом h имеет следующий вид:

$$h_{new} = h \cdot \min \left(facmax, \max \left(facmin, fac \cdot \left(\frac{tol}{err} \right)^{\frac{1}{p+1}} \right) \right).$$

В реализации были взяты следующие значения фактор-параметров:

$$facmax = 1.5, \quad facmin = 0.7, \quad fac = 0.98.$$

Оценка глобальной погрешности производится по формуле:

$$\begin{aligned} \delta(x_{i+1}) &= r_i + \delta(x_i) e^{L_i}, \quad \delta(x_0) = \delta(0) = 0, \\ L_i &= \int_{t_i}^{t_{i+1}} l(\tau) d\tau, \\ l(\tau) &= \mu \left(\frac{J + J^T}{2} \right). \end{aligned}$$

Здесь $r_i = err$ — локальная ошибка на i -м шаге. Под $\mu(A)$, где A — произвольная матрица, понимается максимальное собственное значение этой матрицы; J — матрица Якоби системы ОДУ. В нашем случае:

$$J = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \quad J^T = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \implies \frac{1}{2} \cdot (J + J^T) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Следовательно, $l(\tau) \equiv 0$, $L_i \equiv 0$, $\delta(x_{i+1}) = r_i + \delta(x_i)$.

Числа Рунге R_x , R_z :

$$R_x = \left| \frac{x_{10^{-7}} - x_{10^{-9}}}{x_{10^{-9}} - x_{10^{-11}}} (T) \right| \approx 100^{\frac{s}{s+1}} = 100^{\frac{7}{8}} \approx 56.234.$$

Результаты работы программы:

T	tol	N_{steps}	h	$ \tilde{x}(T) - x(T) $	$ \tilde{z}(T) - z(T) $
100pi	$1 \cdot 10^{-7}$	1946	0.161394	0.16	$1.32 \cdot 10^{-2}$
100pi	$1 \cdot 10^{-9}$	4888	0.0642603	$5.38 \cdot 10^{-2}$	$1.45 \cdot 10^{-3}$
100pi	$1 \cdot 10^{-11}$	12279	0.025583	$2.53 \cdot 10^{-2}$	$3.2 \cdot 10^{-4}$
1000pi	$1 \cdot 10^{-7}$	19465	0.161397	0.11	$6.02 \cdot 10^{-3}$
1000pi	$1 \cdot 10^{-9}$	48888	0.0642603	$3.41 \cdot 10^{-2}$	$5.83 \cdot 10^{-4}$
1000pi	$1 \cdot 10^{-11}$	122799	0.025583	$2.51 \cdot 10^{-2}$	$3.16 \cdot 10^{-4}$
10000pi	$1 \cdot 10^{-7}$	194635	0.161424	0.2	$2.1 \cdot 10^{-2}$
10000pi	$1 \cdot 10^{-9}$	488885	0.0642604	$2.98 \cdot 10^{-3}$	$1.39 \cdot 10^{-5}$
10000pi	$1 \cdot 10^{-11}$	1227999	0.025583	$2.3 \cdot 10^{-2}$	$2.65 \cdot 10^{-4}$
100000pi	$1 \cdot 10^{-7}$	1944728	0.161695	0.18	$2.54 \cdot 10^{-2}$
100000pi	$1 \cdot 10^{-9}$	4888808	0.0642615	$4.76 \cdot 10^{-2}$	$1.23 \cdot 10^{-3}$
100000pi	$1 \cdot 10^{-11}$	12279998	0.025583	$6.21 \cdot 10^{-4}$	$1.15 \cdot 10^{-6}$

Числа Рунге:

T	R_x	R_z
100pi	3.79949	10.4407
1000pi	8.29864	20.382
10000pi	9.81161	83.8008
100000pi	2.77537	19.6846

Оценка глобальной погрешности:

T	tol	δ_{glob}
100pi	$1 \cdot 10^{-7}$	$1.72357 \cdot 10^{-4}$
100pi	$1 \cdot 10^{-9}$	$4.33952 \cdot 10^{-6}$
100pi	$1 \cdot 10^{-11}$	$1.18496 \cdot 10^{-7}$
1000pi	$1 \cdot 10^{-7}$	$1.72418 \cdot 10^{-3}$
1000pi	$1 \cdot 10^{-9}$	$4.33166 \cdot 10^{-5}$
1000pi	$1 \cdot 10^{-11}$	$1.09753 \cdot 10^{-6}$
10000pi	$1 \cdot 10^{-7}$	0.01724
10000pi	$1 \cdot 10^{-9}$	$4.33085 \cdot 10^{-4}$
10000pi	$1 \cdot 10^{-11}$	$1.08879 \cdot 10^{-5}$
100000pi	$1 \cdot 10^{-7}$	0.17227
100000pi	$1 \cdot 10^{-9}$	$4.33072 \cdot 10^{-3}$
100000pi	$1 \cdot 10^{-11}$	$1.08791 \cdot 10^{-4}$