Численное исследование уравненения Бонгоффера – Ван дер Поля.

Оглавление

Постановка задачи

Дано:

$$\begin{cases} x' = -a(\frac{x^3}{3} - x) + ay, \\ y' = -x - by + c, \\ x(0) = 2, \quad y(0) = 0 \end{cases}$$

Здесь $1 \leq a \leq 10^3$, $0 < c < 1$.

Задача: Провести исследование поведения решений в зависимости от значений «большого» параметра a.

Методы решения

Для поиска первообразных было выбрано пять численных метода: *явный метод Эйлера*, *явный метод Рунге-Кутты 4го и 5го порядка*, *неявный метод Эйлера 1го порядка* и *неявный метод Рунге-Кутты 2го порядка (приближение трапецией)*.

Явный метод Эйлера

Имеем отрезок времени $[T_0, T_{\text{fin}}]$. Разобьем его точками $t_0, t_1, \ldots, t_{\text{n}}$, где $t_1 = T_0$, $t_n = T_{\text{fin}}$ и $\tau = t_i - t_{i-1}$ Для каждой i-ой точки найдем (x_{i}, y_{i}) ($0 \leqslant i \leqslant n$). Имеем $\frac{x_i - x_{i-1}}{\tau} = x(x_{i-1}, y_{i-1})$ и $\frac{y_i - y_{i-1}}{\tau} = y(x_{i-1}, y_{i-1})$ ($1 \leqslant i \leqslant n$) Получим уравнения:

$$\text{x}_i = \begin{cases} x_0 & \text{, при } i = 0 \\ x(x_{i-1}, y_{i-1})\tau + x_{i-1} & \text{, иначе} \end{cases}$$

и

$$\text{y}_i = \begin{cases} y_0 & \text{, при } i = 0 \\ y(x_{i-1}, y_{i-1})\tau + y_{i-1} & \text{, иначе} \end{cases}$$

Применение его к данной задаче:

$$\mathrm{x}_i = \begin{cases} 2 & \text{, при } i = 0 \\ (-a\left(\frac{x_{i-1}^3}{3} - x_{i-1}\right) + ay_{i-1})\tau + x_{i-1} & \text{, иначе} \end{cases}$$

и

$$\mathrm{y}_i = \begin{cases} 0 & \text{, при } i = 0 \\ (-x_{i-1} - by_{i-1} + c)\tau + y_{i-1} & \text{, иначе} \end{cases}$$

По этим значениям и будем строить график.

Неявный метод Эйлера

Неявный метод Эйлера, как и явный, имеет первый порядок, но отличается от последнего тем, что для его применения требуется решить систему уравнений:

$$\begin{cases} x_i = x(x_i, y_i)\tau + x_{i-1} \\ y_i = y(x_i, y_i)\tau + x_{i-1} \end{cases}$$

Применений к нашей задаче:

$$\begin{cases} x_i = (-a(\frac{x_i^3}{3} - x_i) + ay_i)\tau + x_i \\ y_i = (-x_i - by_i + c)\tau + y_i \end{cases}$$

Будем численно искать решение этой системы с помощью *метода Ньютона*. С помощью него будем искать x_i. Рассмотрим функцию
$f(x_i) = -x_i + (-a\left(\frac{x_i^3}{3} - x_i\right) + ay_i)\tau_i + x_i = 0$. Суть метода ньютона найти такое $\tilde{x}_n$, такую что $f(\tilde{x}_n) \approx 0$, где n - число итераций алгоритма, а $\tilde{x}_n$ - точка приближения. Зададим начальную точку приближения $\tilde{x}_0 = x_{i-1}$.

$$\tilde{x}_i = \tilde{x}_{i-1} - \frac{f(\tilde{x}_{i-1})}{f'(\tilde{x}_{i-1})} (1 \leqslant i \leqslant n)$$

Таким образом находим n-ую точку приближения и считаем, что она равна x_i. Чем больше итераций будет сделано, тем ближе эта точка будет к настоящему решению, но делать их слишком много тоже не имеет смысла.
После того, как мы найдем таким образом x_i подставляем ее в уравнение для y_i. Так мы получаем точки (x_i, y_i), по ним и строим график.

Метод Рунге-Кутты 4-го порядка

Метод Рунге-Кутты 4-го порядка - явный одношаговый метод. Точность возрастает за счет дополнительных вычислений на отрезке $[t_n, t_n + T]$. Если положить $\frac{dy}{dt} = f(t, y)$, где t - независимая переменная, то алгоритм выглядит так:

$$
\begin{aligned}
&k_1 = f(t_n, y_n),\\
&k_2 = f(t_n + \frac{\tau}{2}, y_n + \frac{\tau}{2}k1),\\
&k_3 = f(t_n + \frac{\tau}{2}, y_n + \frac{\tau}{2}k2),\\
&k_4 = f(t_n + \tau, y_n + \tau k3)\\
&\text{тогда следующее значение } y(t) \text{ считается по формуле:}\\
&y_{n+1} = y_n + \frac{\tau}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

В поставленной задаче фактически три переменных, поэтому, если взять $\frac{dx}{dt} = f_x(t, x, y)$ и $\frac{dy}{dt} = f_y(t, x, y)$, метод примет следующий вид:

$$
\begin{aligned}
&k_{y1} = f_y(t_n, y_n, x_n),\\
&k_{x1} = f_x(t_n, y_n, x_n),\\
&k_{y2} = f_y(t_n + \frac{\tau}{2}, y_n + \frac{\tau}{2}k_{y1}, x_n + \frac{\tau}{2}k_{x1}),\\
&k_{x2} = f_x(t_n + \frac{\tau}{2}, y_n + \frac{\tau}{2}k_{y1}, x_n + \frac{\tau}{2}k_{x1}),\\
&k_{y3} = f_y(t_n + \frac{\tau}{2}, y_n + \frac{\tau}{2}k_{y2}, x_n + \frac{\tau}{2}k_{x2}),\\
&k_{x3} = f_x(t_n + \frac{\tau}{2}, y_n + \frac{\tau}{2}k_{y2}, x_n + \frac{\tau}{2}k_{x2}),\\
&k_{y4} = f_y(t_n + T, y_n + \tau k_{y3}, x_n + \tau k_{x3}),\\
&k_{x4} = f_x(t_n + T, y_n + \tau k_{y3}, x_n + \tau k_{x3}),\\
&\text{соответственно:}\\
&y_{n+1} = y_n + \frac{\tau}{6}(k_{y1} + 2k_{y2} + 2k_{y3} + k_{y4}),\\
&x_{n+1} = x_n + \frac{\tau}{6}(k_{x1} + 2k_{x2} + 2k_{x3} + k_{x4})
\end{aligned}
$$

Метод Рунге-Кутты-Фельберга 5-го порядка

Методы из семейства Рунге-Кутты могут быть представлены через таблицу Бутчера

$$
\begin{array}{c|cccc}
c_1 & a_{11} & a_{12} & \dots & a_{1s} \\
c_2 & a_{21} & a_{22} & \dots & a_{2s} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\
\hline
 & b_1 & b_2 & \dots & b_s
\end{array}
$$

Явные методы имеют общий вид

$$
\begin{aligned}
&k_1 = f(t_n, y_n), \\
&k_2 = f(t_n + c_2 h, y_n + h(a_{21} k_1)), \\
&k_2 = f(t_n + c_2 h, y_n + h(a_{21} k_1)), \\
&\qquad\qquad \vdots \\
&k_i = f(t_n + c_i h, y_n + \tau \sum_{j=1}^{i-1} a_{ij} k_j), \\
&y_{n+1} = y_n + \tau \sum_{i=1}^{s} b_i k_i
\end{aligned}
$$

Так как им соответствует ступенчатая таблица (вот почему суммирование до $i-1$), то есть их применение не требует решения системы уравнений. Поэтому легко реализовать универсальную функцию, которая применяет метод по формулам, используя коэффициенты из таблицы.

Таблиц для метода одного порядка может быть несколько, но они должны удовлетворять некоторым условиям.

Так выглядит одна из таблиц Бутчера для явного метода Рунге-Кутты-Фельберга:

0						
$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$				
$\frac{12}{13}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$			
1	$\frac{1}{4}$	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$	
$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	
	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$

Неявный метод 2-го порядка

Для неявных методов, конечно, тоже существуют таблицы Бутчера. Но так как они не имеют нижнетреугольного вида, общая формула вычисления немного меняется:

$$k_i = f(t_n + c_i h, y_n + \tau \sum_{j=1}^{s} a_{ij} k_j) \qquad \text{(суммирование до } s\text{)}$$

В проекте использовался *метод Кранка-Никольсона* — правило трапеции, только в неявной форме. Таблица Бутчера для него следующая

0	a_{11}	0	0
1	a_{21}	$\frac{1}{2}$	$\frac{1}{2}$
	$\frac{1}{2}$	$\frac{1}{2}$	

$k_{y1} = f_y(t_n, y_n, x_n)$,
$k_{x1} = f_x(t_n, y_n, x_n)$,
Теперь требуется решить систему уравнений для двух неизвестных: k_{x2}, k_{y2}

$$\begin{cases} k_{y2} = f_y(t_n + \frac{\tau}{2}, y_n + \frac{\tau}{2}k_{y1} + \frac{\tau}{2}k_{y2}, x_n + \frac{\tau}{2}k_{x1} + \frac{\tau}{2}k_{x2}), \\ k_{x2} = f_x(t_n + \frac{\tau}{2}, y_n + \frac{\tau}{2}k_{y1} + \frac{\tau}{2}k_{y2}, x_n + \frac{\tau}{2}k_{x1} + \frac{\tau}{2}k_{x2}) \end{cases}$$

Система уравнений в проекте решается с помощью модуля `SciPy` языка `Python`, конкретно, функцией `scipy.optimize.root(fun, x0, jac,`

`method)`. Эта функция решает уравнение с помощью численных методов, принимая на вход

fun — функции, которые обращаются в ноль при подстановке корней

x0 — предпологаемое решение, от которого отталкивается метод

jac — матрица Якоби для функций из fun

method — метод, которым будет решаться уравнение, был выбран `hybr`

Адаптивный шаг

В задачах такого типа зачастую шаг делают не фискированным, а адаптивным, то есть меняющимся во время работы алгоритма. Делается это от того, что жесткие задачи содержат участки быстрого изменения решения, поэтому при взятия большого шага будет значительно терятся точность. Если же брать постоянный маленький шаг алгоритм будет работать слишком долго. Мы также решили реализовать адапативный шаг.

Мы имеешь систему ОДУ, вида

$$\frac{f_i}{dt} = f_i(\mathbf{u}, t) \qquad (1 \leqslant i \leqslant 2)$$

Возьмем новый аргумент - длину дуги интегральной кривой:

$$dl = \rho dt, \rho = \left(1 + \sum f_i^2\right)^{1/2},$$

тогда

$$\frac{f_i}{dl} = F_i(\mathbf{u}), F_i = f_i \rho^{-1} (1 \leqslant i \leqslant 2)$$

Будем выбирать шаг в зависимости от кривизны интегральной кривой, по формуле:

$$\tau = \frac{\tau_0}{(1 + L^{0.5/\nu}(\chi, \chi)^{0.25/\nu})^\nu}$$

Здесь L - полный промежуток интегрирования оп l, χ - кривизна, T_0 - шаг на почти прямом участке интегральной кривой. Численные расчеты показали, что для сверхжестких задач целесообразно выбирать $\nu = 2$, а для задач меньшей жесткости – $\nu = 1$. Мы взяли $\nu = 1$. Таким образом мы строим сетку, начиная с T_0. Делается это одновременно с вычислением решения, поэтому она будеи под него адапитирована. $\chi = T * prev(\mathbf{F} - \mathbf{F} * \mathbf{prev})$, где $T * prev$ - предыдущий шаг. Алгоритмы от применения адаптивного шага не меняются, меняется только сетка точек, по которым мы строим график.

Проверка корректности методов

1ая тестовая задача (не жесткая)

Для задачи

$$

x=y

$$

Численные методы показывают следующую погрешность

Явный метод Эйлера...

2ая тестовая задача (жесткая)

Наша вторая тестовая задача является жесткой. Общепринятого математического определения жестких ОДУ нет, однако у них есть важное свойство, явные методы являются крайне не устойчивыми при решении ими таких задач. Вот сама система:

$$x' = \lambda \left[\cos^2(t)\sin(t) + 2\cos(t) - (2 + xy)x\right] - y$$
$$y' = x + y - \sin(t)$$

Ее жесткость определяется фиксированным параметром λ. Эта задача имеет точное решение в замкнутой форме:

$$x = \cos(t)$$
$$y = \sin(t)$$

Для жесткой задачи оказалось, что явные методы крайне неустойчивы и все имеют погрешность = 1 И чем больше показатель жесткости, тем хуже ситуация

Вывод: анализ задачи лучше производить с помощью неявного метода

Применение методов к поставленной задаче, анализ результатов

Сравним график x от t полученный методами явного 5го порядка и неявного 1го - получим одно и то же, как для прошлой задачи В качестве проверки приведем графики модуля SciPy

В связи со сказанным выше исследования проведем с помощью неявного метода Эйлера