Санкт-Петербургский государственный университет
Факультет прикладной математики — процессов управления
Кафедра информационных систем

Моисеев Владислав Игоревич

Выпускная квалификационная работа бакалавра

Методы Рунге — Кутты для дифференциальных уравнений с запаздывающим аргументом нейтрального типа в MATLAB

Направление 010300
Фундаментальная информатика и информационные технологии

Заведующий кафедрой,
доктор физ.-мат. наук,
профессор — Олемской И. В.

Научный руководитель,
кандидат физ.-мат. наук,
доцент — Ерёмин А. С.

Санкт-Петербург
2017

Содержание

Введение

Дифференциальные уравнения с запаздывающим аргументом (ДУЗА), также известные как уравнения с последействием, широко применяются в математическом моделировании процессов во многих отраслях науки, например, при моделировании дорожного траффика, экономических процессов, систем управления с обратными связями, динамики популяций, иммунологических процессов в организме, распространения эпидемий, нелинейных оптических сред, материалов с памятью, динамики жидкостей, процессов в химических и ядерных реакторах, в линиях электропередачи, в двигателях сгорания и очень многих других процессов в биологии, медицине, механике, физике, технике [1, 2]. Их теоретическое исследование началось достаточно давно (см., например, классические работы Эльсгольца и Норкина [3] и Колмановского и Носова [4], или более поздние фундаментальную работы Хейла и Вердун Люнеля [5] и Колмановского и Мышкиса [6]).

Определение 1. *Дифференциальным уравнением с запаздывающим аргументом* называется уравнение вида:

$$\frac{dy}{dt} = f\big(t, y(t), y(t-\tau_1), \ldots, y(t-\tau_n)\big), \quad \tau_i > 0, \quad i = 1, ..., n. \tag{1}$$

Для постановки начальной задачи необходимо задавать не только значение решения в начальной точке t_0, но некоторую функцию $\varphi(t)$, $t \in [t_0 - \tau, t]$, $\tau = \max\limits_i \tau_i$, называемую «предысторией», определяющую поведение решения на целом интервале в прошлом.

Первые попытки построить численные методы решения ДУЗА были предприняты достаточно давно, но в недавнее время они развиваются особенно активно в связи с ростом производительности ЭВМ. Основные результаты из этой темы собраны в монографии Беллена и Ценнаро [7] и их более поздней статье с ещё двумя соавторами [8].

Так как при решении дифференциальных уравнений с запаздыванием в общем случае являются кусочно-гладкими функциями, применение многошаговых методов довольно затруднительно. Большинство применяемых интеграторов основано на одношаговых методах, например популярные программы RETARD и RADAR Хайрера и Гуглиэлми [9], программа Фельдстейна и Невеса [10], методы из работ Энрайта и Ху [11] и Хаяши [12].

Среди уравнений с последействием выделяют класс ДУЗА так называемого *нейтрального типа.*

Определение 2. *Дифференциальным уравнением с запаздывающим*

аргументом нейтрального типа (ДУЗАНТ) называется уравнение вида:

$$\frac{dy}{dt} = f\big(t, y(t), y(t-\tau_1), \ldots, y(t-\tau_n), y'(t-\sigma_1), \ldots, y'(t-\sigma_m)\big), \quad \tau_i > 0, \quad i = 1, ..., n, \quad \sigma_j > 0, \quad j = 1, ...m. \tag{2}$$

Обратим внимание, что в обоих случаях (1) и (2) запаздывания τ_i и σ_j могут быть функциями от времени t и самого решения $y(t)$. В данной работе мы будем рассматривать только ситуации, в которых $\tau_i \geqslant \bar{\tau} > 0$ $i = 1, ..., n$ и $\sigma_j \geqslant \bar{\sigma} > 0$, $j = 1, ...m$ при любых встречающихся в процессе решения значениях t и $y(t)$. При этом будем считать, что нижние границы запаздываний $\bar{\tau}$ и $\bar{\sigma}$ достаточно велики, чтобы шаг применяемых численных методов мог быть выбран не превышающим их.

Уравнения нейтрального типа также часто возникают в моделях инфекционных заболеваний, популяционной динамике, физиологии, фармацевтической и химической кинетике, навигации водных и воздушных судов, задачах управления, моделировании линий электропередач и др. (см., напр., [13]).

ДУЗАНТ изучены куда хуже, чем ДУЗА. Адаптация к ним численных методов, применимых к решению обыкновенных дифференциальных уравнений (ОДУ), или разработка новых специальных методов сложнее, чем в случае уравнений запаздывающего типа. Так, например, метод шагов Беллмана [14] приводит к более сложным уравнениям, а непрерывные методы Рунге — Кутты [7] требуют не только построения довольно точного решения внутри шага, но и достаточно точного приближения его производной.

Мы рассмотрим реализацию процедуры для решения ДУЗАНТ в MATLAB. Покажем, что эта процедура имеет ряд недостатков и предложим альтернативную процедуру, основанную на непрерывном методе Рунге — Кутты, разработанном Оуреном и Ценнаро [15].

В дальнейшей работе будем полагать, что правая часть уравнения (2) достаточно гладкая по всем аргументам, чтобы выполнялись условия существования и единственности решения начальной задачи [5].

Реализация метода Рунге — Кутты для решения ДУЗАНТ в MATLAB

Рассмотрим методы решения дифференциальных уравнений с запаздыванием нейтрального типа средствами MATLAB.

Функция ddesd позволяет находить решение для дифференциальных уравнений с запаздывающим аргументом путём использования «классического» метода Рунге — Кутты 4-го порядка для обыкновенных дифференциальных уравнений и функции lagvals, которая, в свою очередь, возвращает значение запаздывания в конкретной точке. Запаздывание находится исходя из предыстории, если запаздывание $t - \tau < t_0$, и вычисленных ранее приближённых значений решения y в точках сетки. Если запаздывание $t - \tau$ не попадает в точку сетки, а попадает внутрь определённого совершённого ранее шага, то строится интерполяционный полином Эрмита 3-го порядка по значениям y и f в краях этого шага, и по нему находится значение решения в нужной точке $t-\tau$. Данная процедура основана на работе Шампайна [16].

Функция ddensd, теоретическая часть которой также выведена Шампайном [17], позволяет находить решение для дифференциальных уравнений с запаздывающим аргументом нейтрального типа. Устроена она так, чтобы решение находилось процедурой ddesd для измененной функции f, в которой значения $y'(t - \sigma_1)$, $\ldots$, $y'(t - \sigma_m)$ являются параметрами, вычисляемыми с помощью приближенного дифференцирования

$$y'(t - \sigma) = \frac{y(t - \sigma + \Delta t) - y(t - \sigma)}{\Delta t}. \tag{3}$$

Выбирается некий «малый» отступ по времени Δt, находятся значения запаздываний $y(t - \sigma)$ и $y(t - \sigma + \Delta t)$, и вычисляется само приближение к производной. Как мы увидим далее, именно это «малое» Δt и является слабым местом процедуры ddensd.

Управление шагом ведётся в обеих процедурах ddesd и ddensd по оценке невязки [18].

Постановка задачи

Существующий метод для решения дифференциальных уравнений с запаздыванием нейтрального типа в MATLAB имеет ряд проблем, которые могут оказывать сильное влияние на получаемый результат при решении реальных задач. Существенной проблемой является, то что малый отступ по времени Δt в уравнении (3), при нахождении производной запаздывания, не зависит от шага и ограничен некоторой малой величиной MINCHANGE. Таким образом, при уменьшении шага, теряется порядок приближения запаздывающей производной, что влечёт за собой потерю порядка самого метода.

Главной задачей данной работы, является описание метода, которые включает в себя решение данной проблемы, а также, который смог бы полностью заменить существующее решение MATLAB.

Непрерывный метод Рунге — Кутты для решения ДУЗАНТ

Поскольку вопрос об устойчивости численных методов для ДУЗА очень слабо изучен и определить жёсткость [19] задачи в случае уравнений с запаздываниями весьма непросто, применение явных методов широко распространено, и большая часть используемых в настоящее время методов — явные. В частности, процедуры ddesd и ddensd.

Поэтому и мы для решения поставленной задачи выбрали явный непрерывный метод Оурена и Ценнаро [15], являющийся 6-этапным методом Рунге — Кутты 4-го порядка с плотной выдачей и повторным использованием (т. е. фактически, вычисляется только 5 этапных функций на шаг).

Явный метод Рунге — Кутты [20] решения начальной задачи для ОДУ

$$\begin{cases} \dfrac{dy}{dt} = f\big(t, y(t)\big), \\ y(t_0) = y_0 \end{cases} \tag{4}$$

даёт приближение через шаг длины h

$$\begin{aligned} y(t_0 + h) \approx y_1 &= y_0 + h(b_1 k_1 + \ldots + b_s k_s), \\ k_1 &= f(x_0, y_0), \\ k_2 &= f(x_0 + c_2 h, y_0 + h a_{2,1} k_1), \\ k_3 &= f\big(x_0 + c_3 h, y_0 + h(a_{3,1} k_1 + a_{3,2} k_2)\big), \\ &\ldots \\ k_s &= f\big(x_0 + c_s h, y_0 + h(a_{3,1} k_1 + \ldots + a_{s,s-1} k_{s-1})\big). \end{aligned} \tag{5}$$

Число s называется числом этапов, а $a_{2,1}$, $a_{3,1}$, $a_{3,2}$, …, $a_{s,1}$, $a_{s,2}$, $a_{s,s-1}$, …, b_1, …, b_s, c_2, …, c_s — коэффициенты метода.

Определение 3. *Локальным порядком* метода (5) называется натуральное число p, если разложение в ряд Тейлора по степеням h решения $y(t + h)$ и приближения y_1 совпадают до порядка p включительно, т. е.

$$\|y(t + h) - y_1\| = O(h^{p+1}).$$

Глобальным порядком или порядком сходимости метода (5) называется натуральное число p, если глобальная погрешность после N шагов, совершённых методов, удовлетворяет соотношению

$$\|y(t + Nh) - y_N\| = O(h^p),$$

где y_{i+1} получено применением одного шага метода (5) к задаче (4) с из-

меннёными начальной точкой $(t + ih, y_i)$, $i = 0, ..., N - 1$.

Отметим, что для ОДУ с достаточной гладкостью правой части локальный и глобальный порядки совпадают.

Плотная выдача (или, иначе, непрерывный метод) позволяет приблизить любое промежуточное значение на промежутке $[t_0; t_0 + h]$

$$y(t + \theta h) \approx y_1(\theta) = y_0 + h\big(b_1(\theta)k_1 + \ldots + b_s(\theta)k_s\big), \quad \theta \in [0, 1]. \qquad (6)$$

Здесь коэффициенты $b_i(\theta)$ являются полиномами, обеспечивающими необходимый порядок аппроксимации $\theta \in [0; 1]$.

Определение 4. *Дискретным (локальным) порядком* метода (6) называется натуральное число p, если разложение в ряд Тейлора по степеням h решения $y(t+h)$ и приближения y_1 совпадают до порядка p включительно, т. е.

$$\|y(t + h) - y_1\| = O(h^{p+1}).$$

Равномерным (локальным) порядком метода (6) называется натуральное число p, если разложение в ряд Тейлора по степеням h решения $y(t + \theta h)$ и приближения $y_1(\theta)$ совпадают до порядка p включительно, т. е.

$$\|y(t + \theta h) - y_1(\theta)\| = O(h^{p+1}), \quad \theta \in [0, 1].$$

На Рис. 1 изображена таблица Бутчера [15] для непрерывного метода Рунге — Кутты (5). В действительности выполняются всего пять этапов, в связи с тем, что свойство FSAL (First Same As Last) предполагает, что в качестве k_1 на новой итерации можно использовать k_6 из предыдущей итерации. Данное свойство позволяет сэкономить затраты на память и время.

Продифференцировав уравнение (6) по $h\theta$ получим уравнение для приближения производной запаздывающего решения:

$$y'(t + h\theta) \approx \sum_{i=1}^{6} b_i'(\theta)k_i \qquad (7)$$

где $b_i'(\theta) = \frac{db_i'(\theta)}{d\theta}$.

Известен результат [7], что метод решения (2) будет иметь порядок сходимости p, если аппроксимация $y(t - \tau)$ и $y'(t - \sigma)$ осуществится с порядками не ниже $p - 1$, что выполняется для предложенного метода.

0						
$\frac{1}{6}$	$\frac{1}{6}$					
$\frac{11}{37}$	$\frac{44}{1369}$	$\frac{363}{1369}$				
$\frac{11}{17}$	$\frac{3388}{4913}$	$-\frac{8349}{4913}$	$\frac{8140}{4913}$			
$\frac{13}{15}$	$-\frac{36764}{408375}$	$\frac{767}{1125}$	$-\frac{32708}{136125}$	$\frac{210392}{408375}$		
1	$\frac{1697}{18876}$	0	$\frac{50653}{116160}$	$\frac{299693}{1626240}$	$\frac{3375}{11648}$	
	$b_1(\theta)$	$b_2(\theta)$	$b_3(\theta)$	$b_4(\theta)$	$b_5(\theta)$	$b_6(\theta)$

$$
\begin{aligned}
b_1(\theta) &= -\tfrac{866577}{824252}\theta^4 + \tfrac{1806901}{618189}\theta^3 - \tfrac{104217}{37466}\theta^2 + \theta,\\
b_2(\theta) &= 0,\\
b_3(\theta) &= \tfrac{12308679}{5072320}\theta^4 - \tfrac{2178079}{380424}\theta^3 + \tfrac{861101}{230560}\theta^2,\\
b_4(\theta) &= -\tfrac{7816583}{10144640}\theta^4 + \tfrac{6244423}{5325936}\theta^3 - \tfrac{63869}{293440}\theta^2,\\
b_5(\theta) &= -\tfrac{624375}{217984}\theta^4 + \tfrac{982125}{190736}\theta^3 - \tfrac{1522125}{762944}\theta^2,\\
b_6(\theta) &= \tfrac{296}{131}\theta^4 - \tfrac{461}{131}\theta^3 + \tfrac{165}{131}\theta^2.
\end{aligned}
$$

Рис. 1: Таблица Бутчера и непрерывные веса непрерывного метода Рунге — Кутты четвёртого порядка с шестью этапами

Сравнение результатов

С постоянным шагом

Сравним два метода с постоянным шагом на примере дифференциального уравнения построенного на основе одной из тестовых задач из часто использующегося набора задач [21]:

$$\begin{cases} y'(t) = 1 + y - 2y\left(\frac{t}{2}\right)^2 - y'(t-\pi), & t \geqslant 0 \\ Y(t) = \cos t, & t \leqslant 0 \end{cases} \tag{8}$$

с точным решением $y(t) = \cos t$.

На Рис. 2 и Рис. 3 продемонстрирован пример работы видоизменённого ddensd для вышеописанного дифференциального уравнения. В данной функции был убран механизм автоматического управления шага, а решения находились на отрезке $[1;\ 6]$ с различными вариациями постоянного шага. На Рис. 2 можно заметить, что погрешность достигает своего минимума и не может быть меньше заданной величины.

Порядок метода Рунге — Кутты определяется степенью уменьшения ошибки при уменьшении шага. Для того чтобы определить порядок, необходимо построить график зависимости погрешности от шага в логарифмической шкале. Если зависимость параллельна прямой с тангенсом угла наклона равным четырём, соответственно, имеем метод 4-го порядка. Такой график изображен на Рис. 3 для функции ddensd. На нём можно увидеть, что при больших шагах порядок сохраняется, но в какой-то момент шаг становится достаточно малым, и за счёт увеличения ошибки при нахождении запаздывающей производной, теряется порядок самого метода.

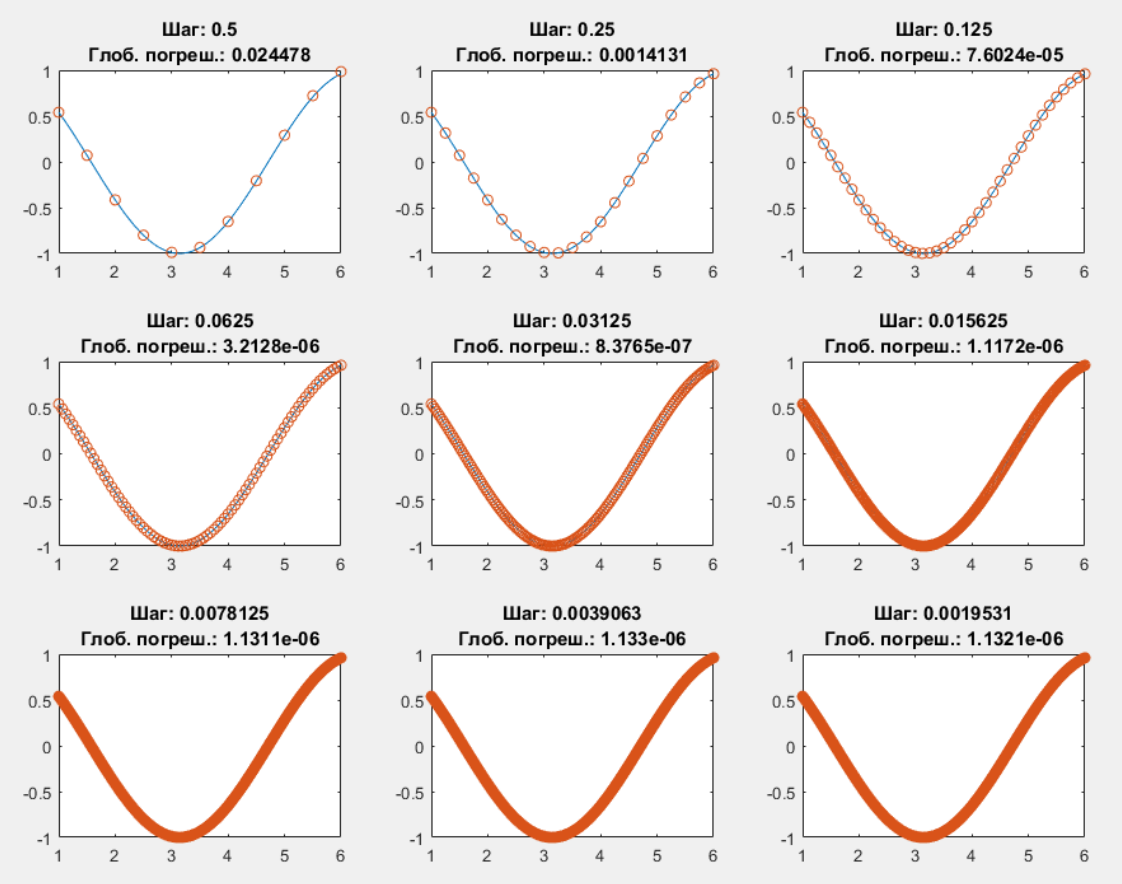

Рис. 2: Пример работы ddensd с постоянным шагом

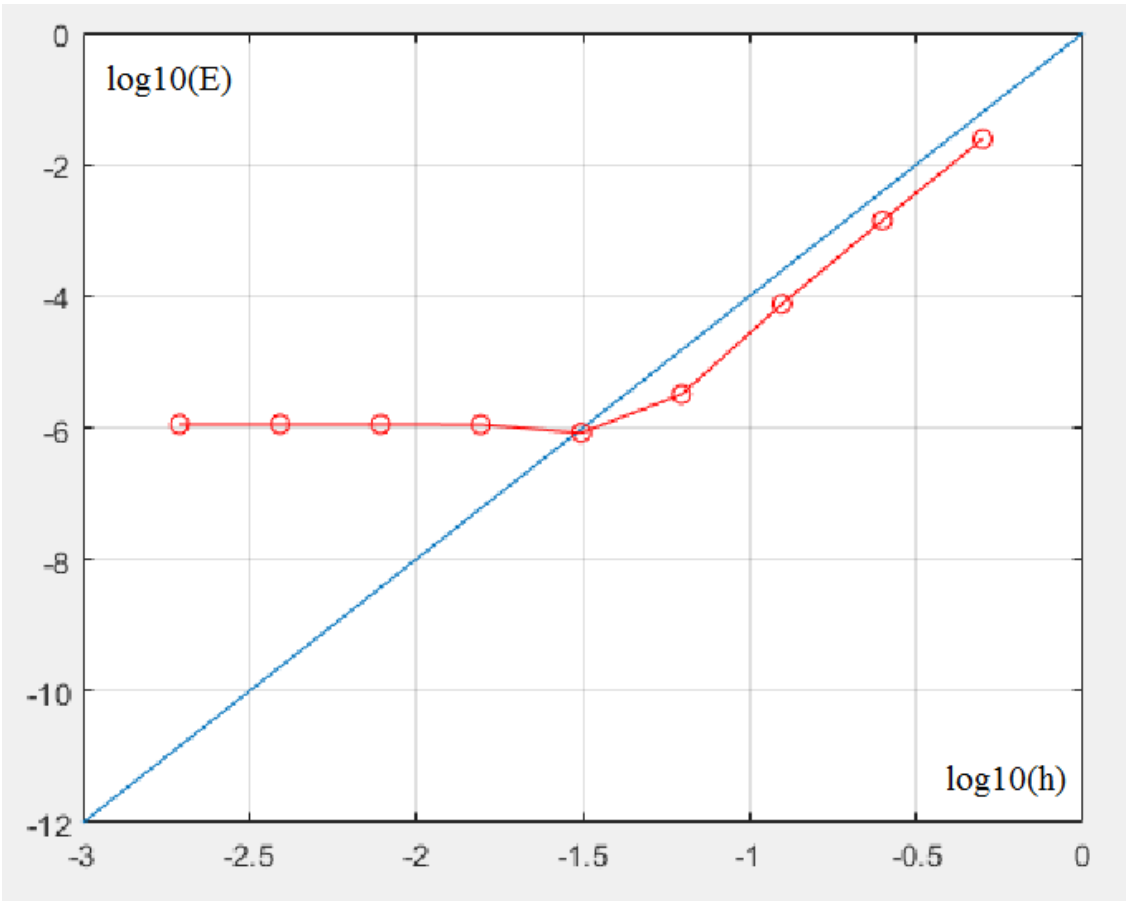

Рис. 3: График логарифмической зависимости погрешности от шага ddensd

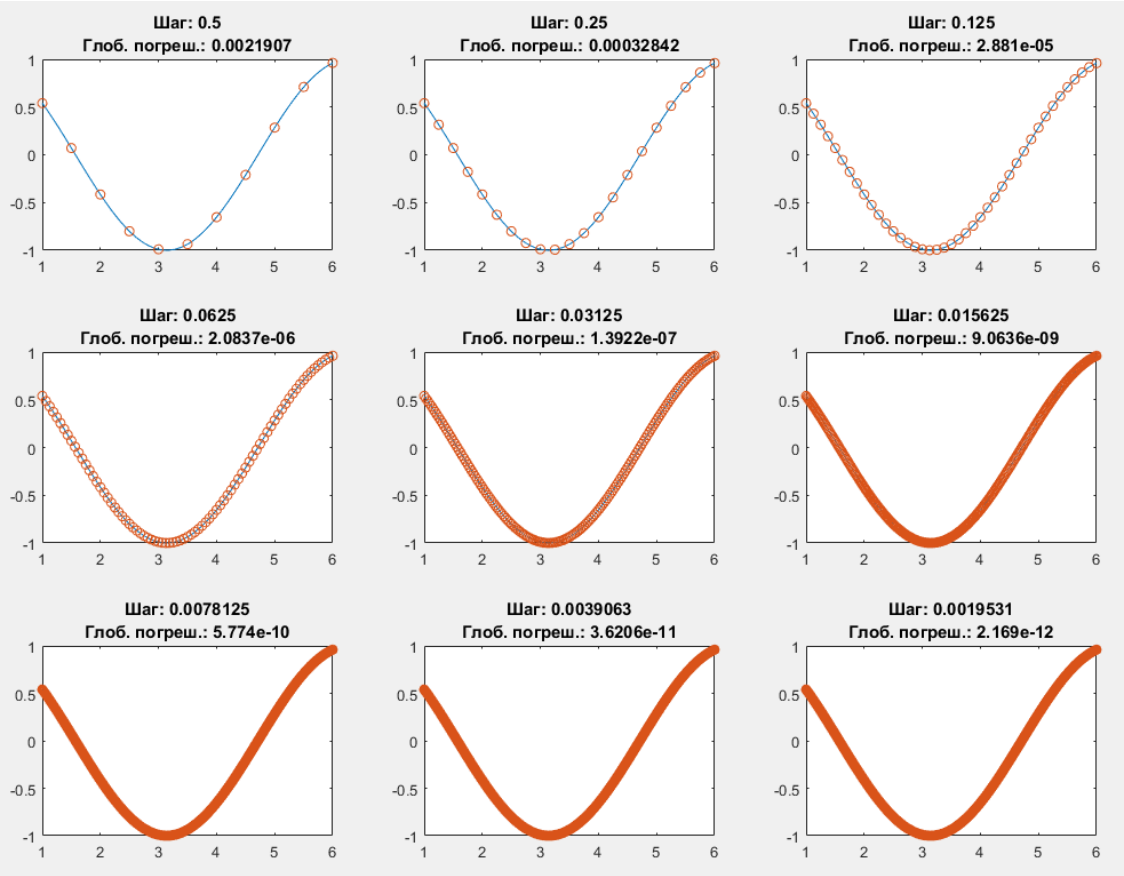

Рис. 4: Пример работы непрерывного метода Рунге — Кутты с постоянным шагом

На Рис. 4 и Рис. 5 продемонстрирован пример работы непрерывного метода Рунге — Кутты с постоянным шагом для того же дифференциального уравнения. Как и для предыдущего метода решения получили на отрезке [1; 6] с такими же шагами. Как видим, данный метод исключает вышеописанные проблемы ddensd: погрешность может быть сколь угодно малой, что ведёт за собой сохранение порядка метода при всевозможных вариаций шагов.

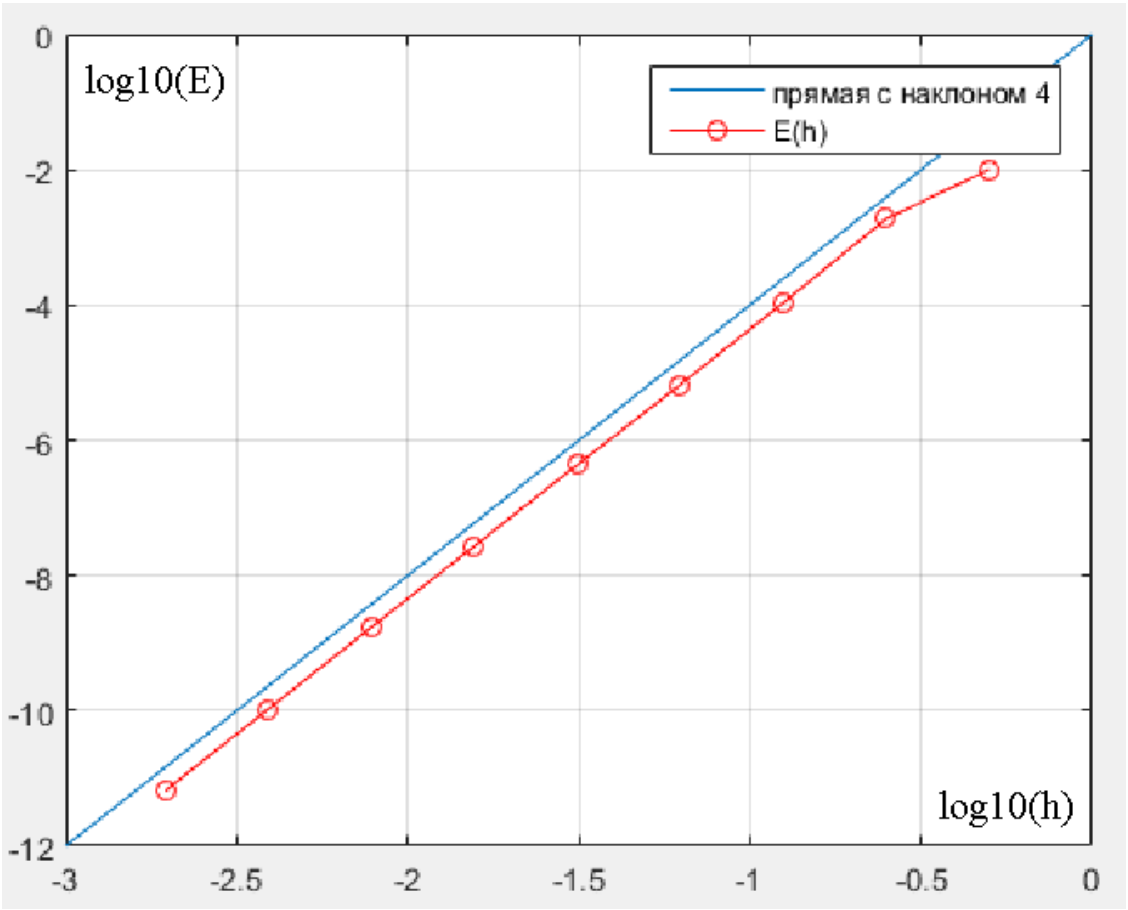

Рис. 5: График логарифмической зависимости погрешности от шага непрерывного метода Рунге — Кутты

С автоматическим управлением шага

Метод автоматически подбирает шаг таким образом, чтобы локальная погрешность была меньше некоторой заданной допустимой погрешности. В итоге получаем результат, вычисленный с нужной для нас точностью.

Автоматическое управление шагом выполняется следующим образом: задаётся допустимая погрешность, выполняется оценка погрешности на текущем шаге, в зависимости от того, меньше или больше допустимая погрешность, чем некоторая оценка, шаг уменьшается или увеличивается определённым образом. Основное различие методов автоматического управления шагом, заключается в выборе оценщика погрешности.

Оценщик погрешности ddensd для вычисления ошибки использует полином Эрмита 3-го порядка. Полином строится по значениям y и f в краях текущего шага. Оценка выполняется следующим образом:

$$err = ypm - ddefun(xm, ym, delayVals, pdelayVals)$$

где ypm - значение производной полинома Эрмита в точке xm, ym - значение полинома Эрмита в точке xm, $ddefun$ - функция возвращающая значение исходного дифференциального уравнения, delayVals и pdelayVals - функции для вычисления запаздывания и запаздывающей производной. После нормировки err, это и будет являться некоторой оценкой погрешности на текущем шаге.

В непрерывном методе Рунге — Кутты автоматическое управление шагом выполняется аналогичным образом.

На Рис. 6 и Рис. 9 продемонстрированы примеры работ ddensd и непрерывного метода Рунге — Кутты с относительной погрешностью RelTol изменяющаяся на интервале [1e-2; 1e-10] для дифференциального уравнения (8). Если посмотреть на пример работы ddensd, видим, как и в предыдущем параграфе, что глобальная погрешность ограничена снизу некоторой величиной. Соответственно не имеет смысла задавать погрешность ниже 1e-05. У непрерывного метода Рунге — Кутты всё иначе, зависимость глобальной и относительной погрешностей имеет порядок 1 на всём наборе RelTol. Это хорошо видно из Рис. 10.

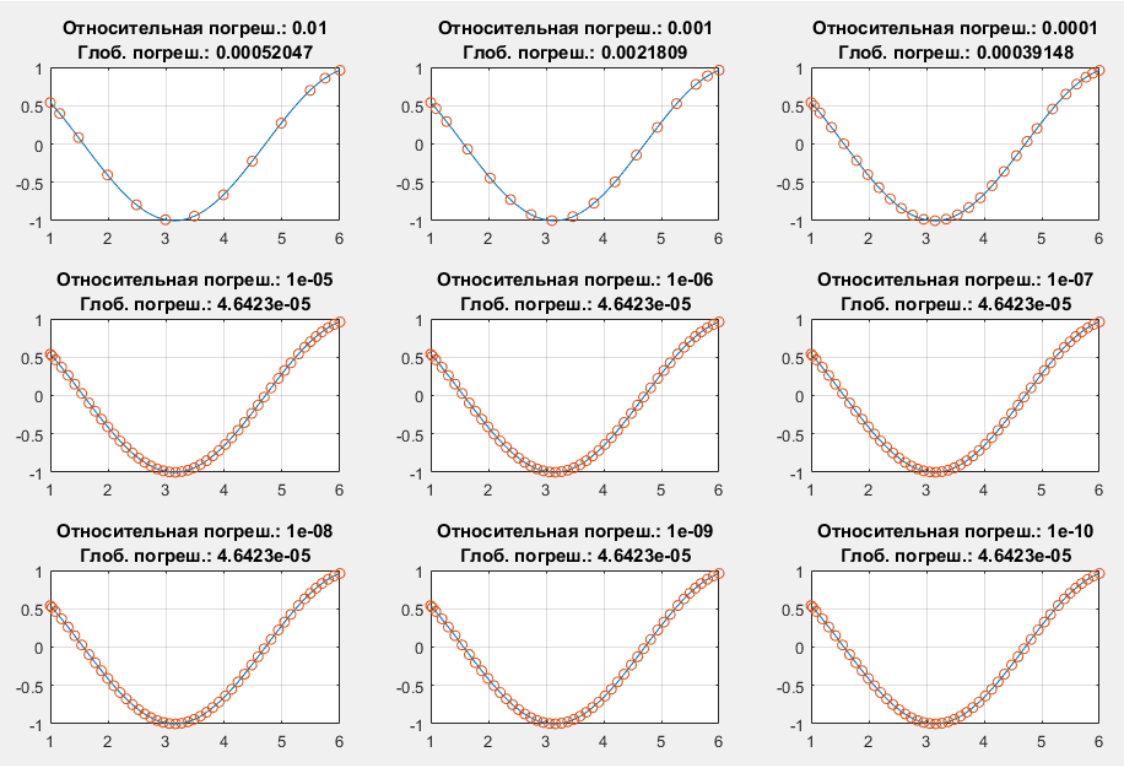

Рис. 6: Пример работы ddensd

Рассмотрим пример со стороны затрат ресурсов на выполнение метода. Введём единицу измерения, по которой будем оценивать затратность метода. Пусть nfeval — количество обращений программы к входной функции дифференциального уравнения. Непрерывный метод Рунге — Кутты более затратный относительно ddensd в связи с тем, что непрерывный метод в ходе одной итерации выполняет 5 этапов, а ddensd всего 4. На Рис. 12 и Рис. 13 представлены таблицы с данными глобальных погрешностей GlErr и количество операций nfevals при заданной относительной погрешности RelTol. Из таблиц видно, что, действительно, ddensd использует меньше ресурсов, но разница настолько мала, что любой современный компьютер не даст её почувствовать пользователю.

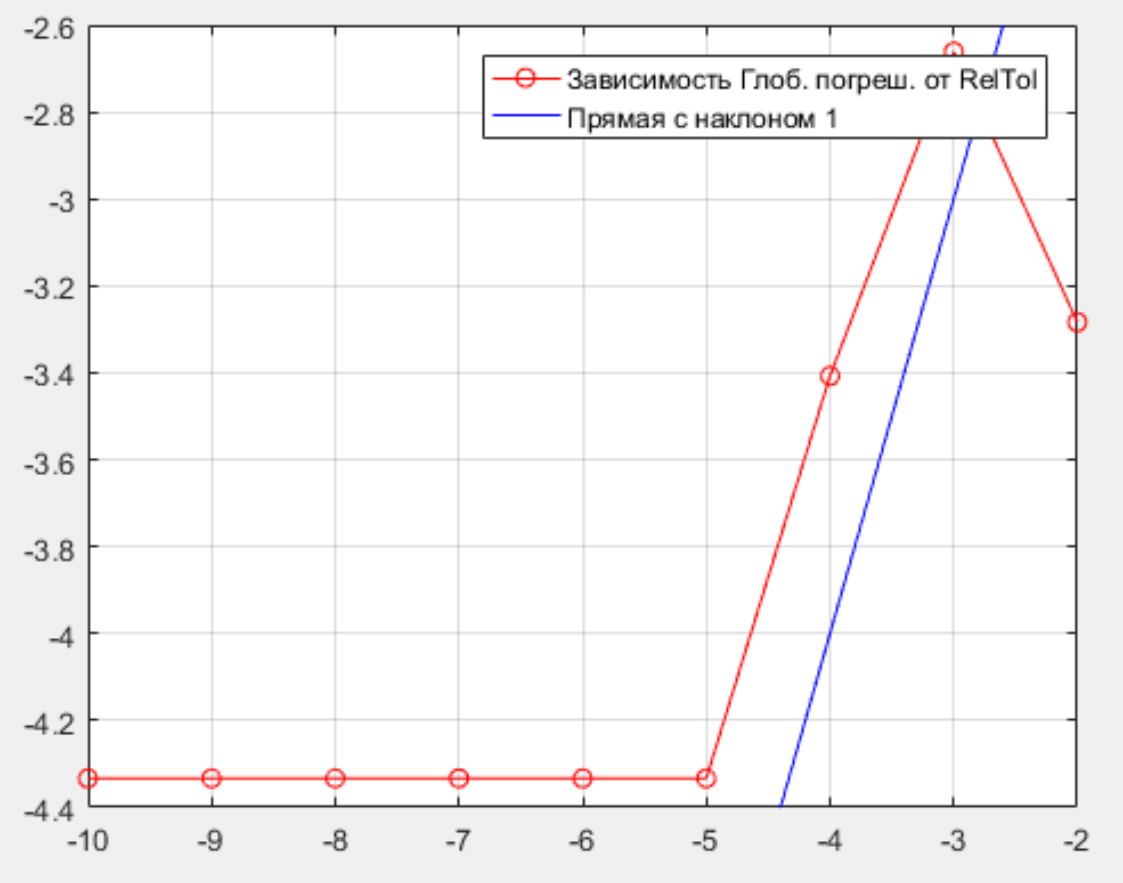

Рис. 7: График логарифмической зависимости глобальной погрешности от относительной погрешности ddensd

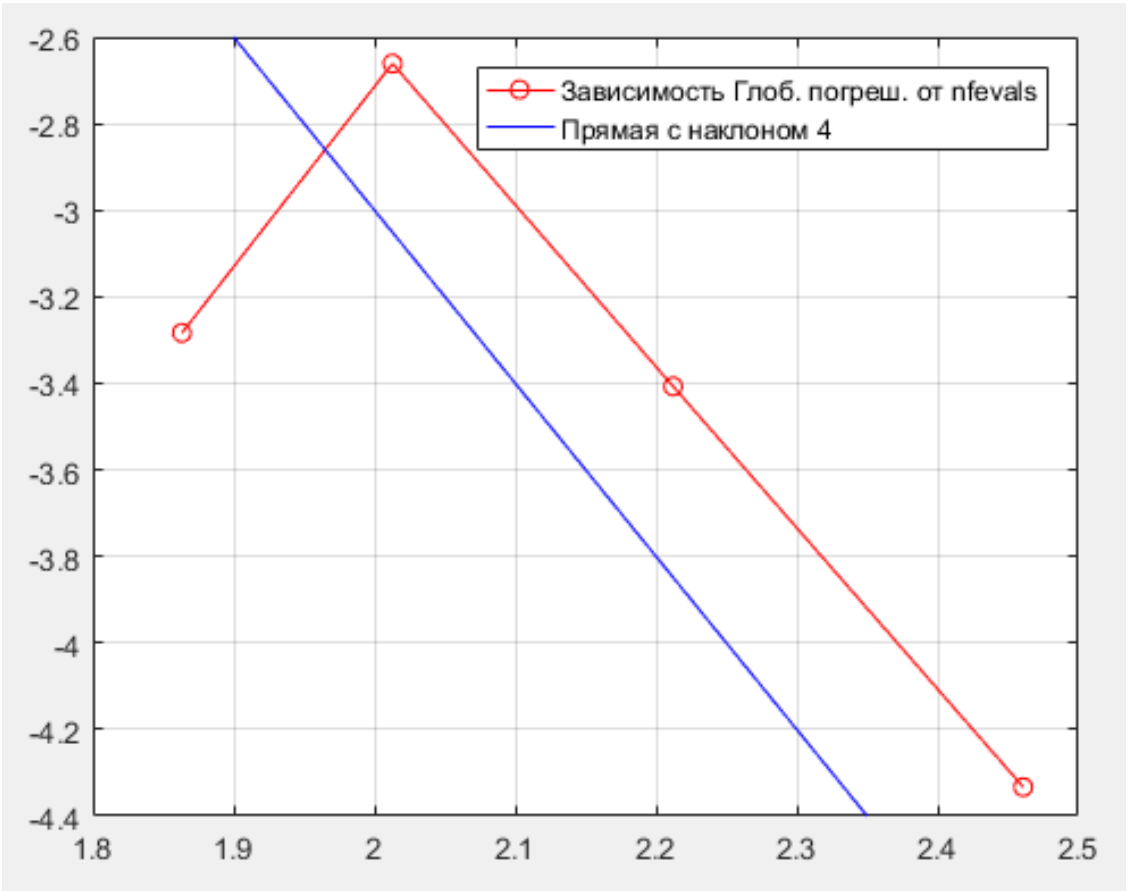

Рис. 8: График логарифмической зависимости глобальной погрешности от nfevals ddensd

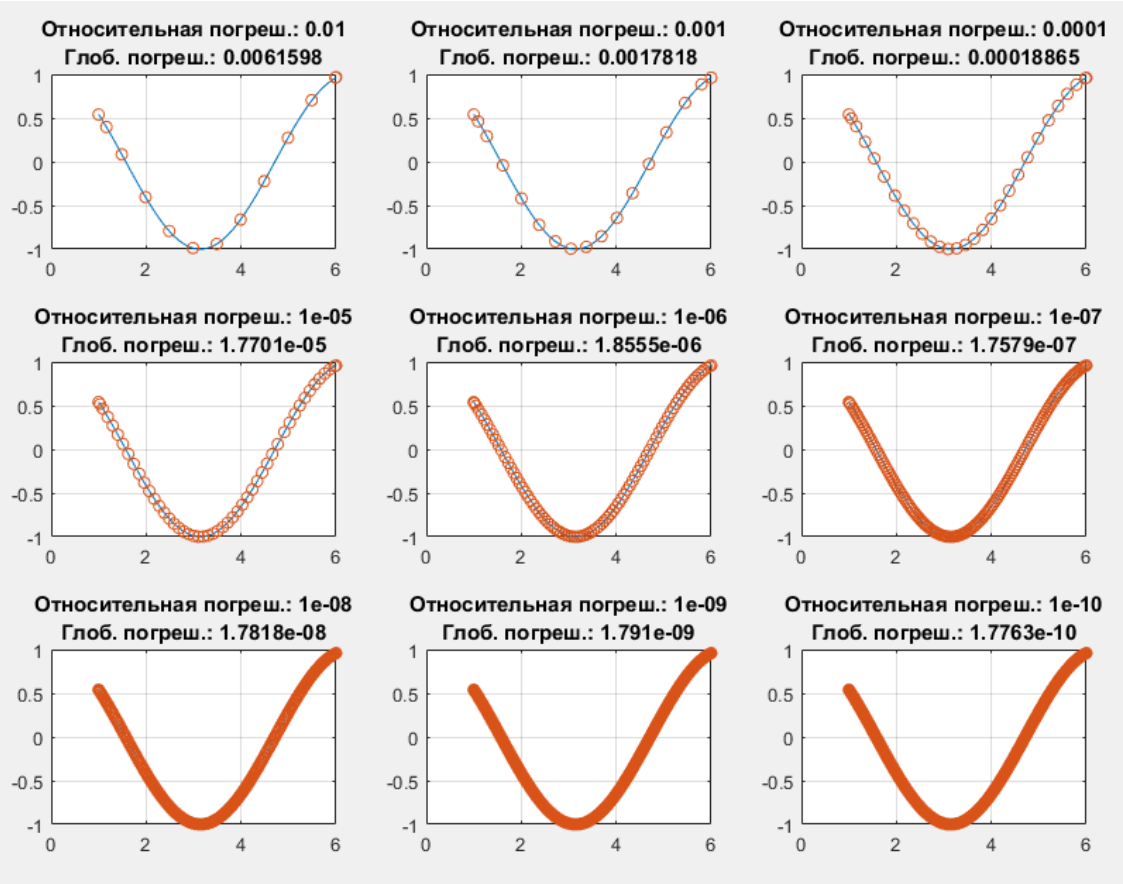

Рис. 9: Пример работы непрерывного метода Рунге — Кутты

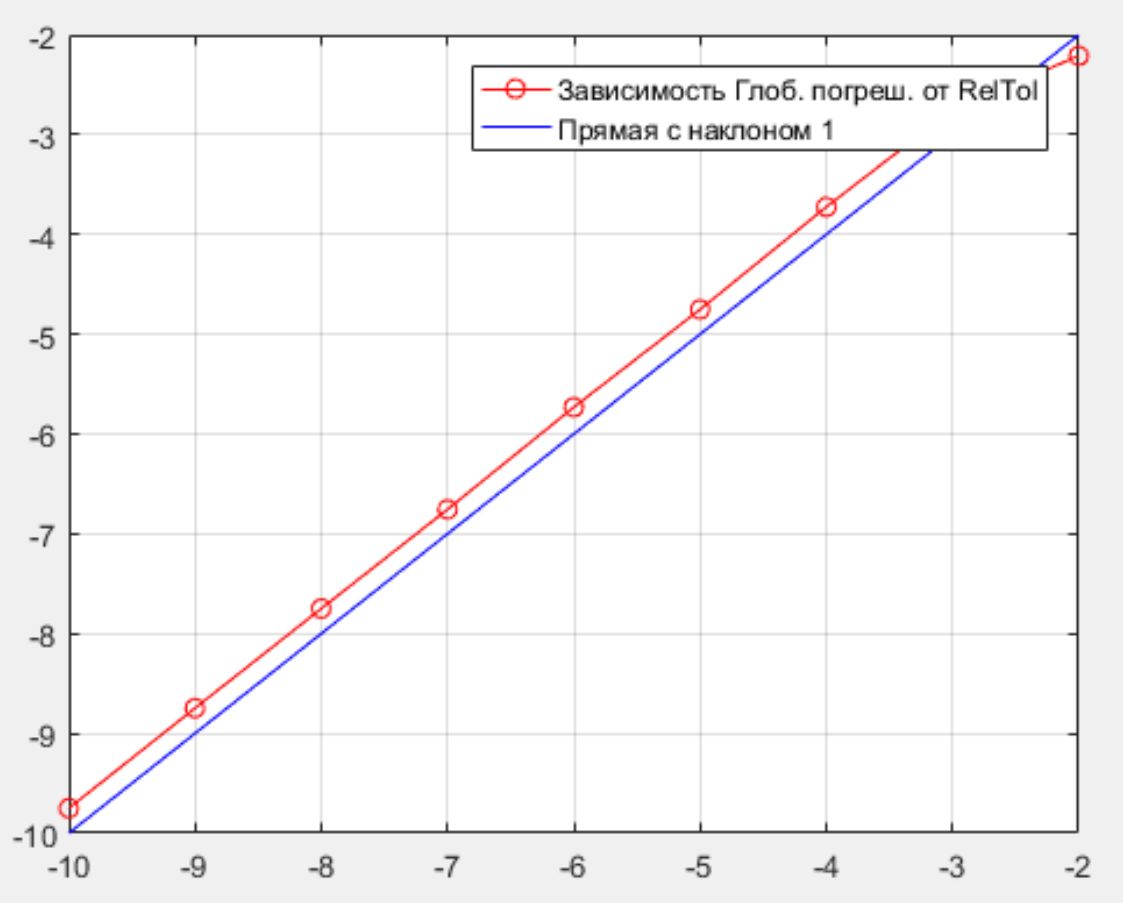

Рис. 10: График логарифмической зависимости глобальной погрешности от относительной погрешности непрерывного метода Рунге — Кутты

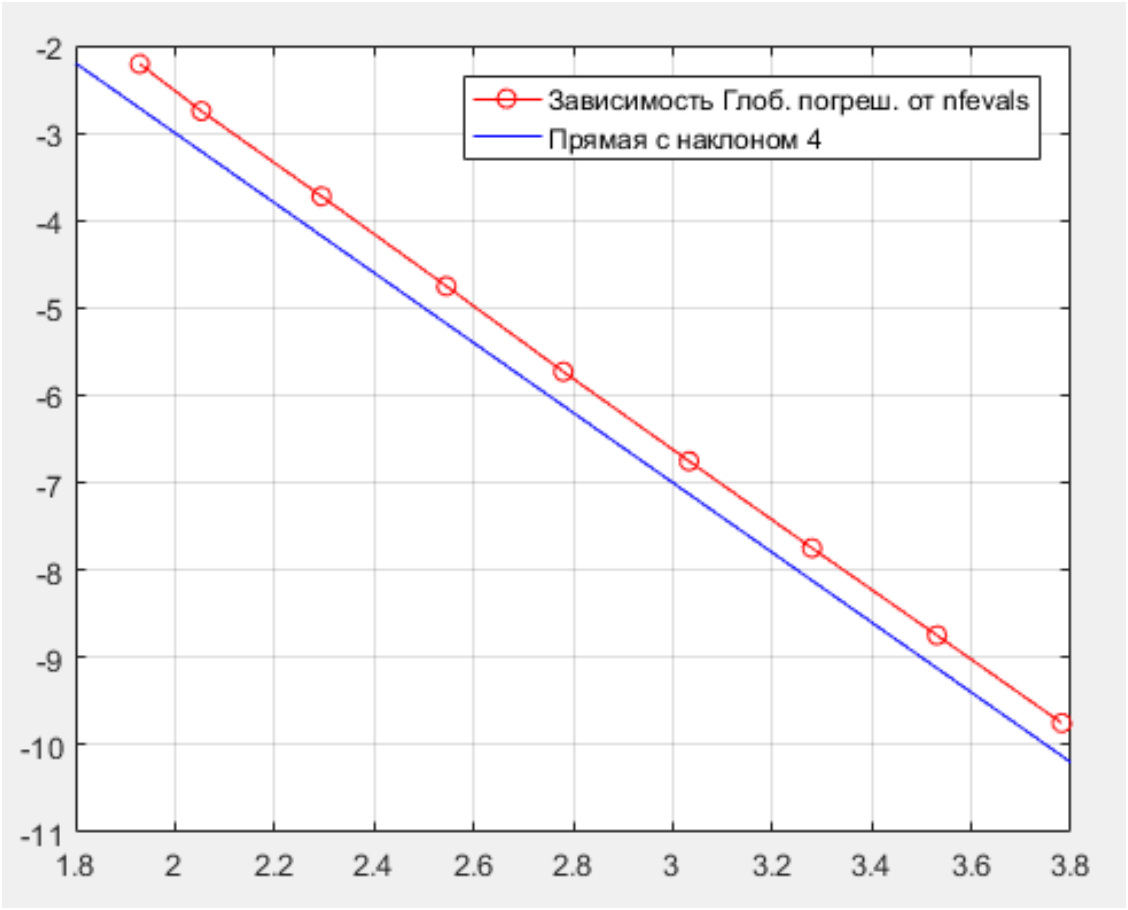

Рис. 11: График логарифмической зависимости глобальной погрешности от nfevals непрерывного метода Рунге — Кутты

RelTol	0.0100	1.0000e-03	1.0000e-04	1.0000e-05	1.0000e-06	1.0000e-07	1.0000e-08	1.0000e-09	1.0000e-10
GlErr	5.2047e-04	0.0022	3.9148e-04	4.6423e-05	4.6423e-05	4.6423e-05	4.6423e-05	4.6423e-05	4.6423e-05
nfevals	73	103	163	289	289	289	289	289	289

Рис. 12: Таблица данных ddensd

RelTol	0.0100	1.0000e-03	1.0000e-04	1.0000e-05	1.0000e-06	1.0000e-07	1.0000e-08	1.0000e-09	1.0000e-10
GlErr	0.0062	0.0018	1.8865e-04	1.7701e-05	1.8555e-06	1.7579e-07	1.7818e-08	1.7910e-09	1.7763e-10
nfevals	85	113	197	351	603	1079	1905	3396	6049

Рис. 13: Таблица данных непрерывного метода Рунге — Кутты

Заключение

Реализация алгоритма, описанного в данной работе, для решения дифференциальных уравнений с запаздывающим аргументом нейтрального типа показывает результаты лучше, нежели алгоритм MATLAB. Также данный алгоритм можно использовать в различных проектах вместо ddensd без переписывания основной программы и без адаптации под новый метод так, как входные и выходные параметры полностью аналогичны ddensd MATLAB.

Список литературы

[1] Erneux T. Applied delay differential equations — NY: Springer Science+ Business Media, LLC, 2009. 204 с.

[2] Smith H. An introduction to delay differential equations with applications — NY: Springer Science+ Business Media, LLC, 2011. 172 с.

[3] Эльсгольц Л. Э., Норкин С. Б. Введение в теорию дифференциальных уравнений с отклоняющимся аргументом — М.: Главная редакция физико-математической литературы изд-ва «Наука», 1971. 296 с.

[4] Kolmanovskii, V.B., Nosov V.R. Stability of functional differential equations — London: Academic Press, 1986. 217 с.

[5] Hale J.K., Verduyn Lunel S.M. Introduction to functional differential equations — NY: Springer Science+ Business Media, LLC, 1993. 447 с.

[6] Kolmanovskii V., Myshkis A. Introduction to the theory and applications of functional differential equations —Dordrecht: Kluwer Academic Publishers, 1999. 648 с.

[7] Bellen A., Zennaro M. Numerical Methods for delay differential equations. New-York: Oxford University Press, 2003. 395 с.

[8] Bellen A., Maset S., Zennaro M., Guglielmi N. Recent trends in the numerical solution of retarded functional differential equations // Acta Numerica, 2009, pp. 1–110.

[9] Fortran and MATLAB codes. Персональный сайт Э. Хайрера. URL: http://www.unige.ch/ hairer/software.html (Дата обращения 24.05.2017).

[10] Feldstein A., Neves K.W. High order methods for state-dependent delay differential equations with nonsmooth solutions // SIAM J. Numer. Anal., 1984, vol.21, no. 5, pp. 844–863.

[11] Enright W.H., Hu M. Interpolating RKs for vanishing delay differential equations // Computing, 1995, vol. 55, pp. 223–236.

[12] Hayashi H. Numerical solution of retarded and neutral delay differential equations using continuous Runge–Kutta methods — PhD Thesis, Department of Computer Science, University of Toronto, Toronto, Canada, 1996.

[13] Kuang Y. Delay differential equations with applications in population dynamics. Mathematics in science and engineering, 191. — Boston: Academic Press, Inc., 1993. 398 с.

[14] Bellman R. On the computational solution of differential-difference equations // J. Math. Anal. Appl., 1961, vol. 2, pp. 108–110.

[15] Owren B., Zennaro M. Derivation of efficient continuous explicit Runge–Kutta methods // SIAM J. Sci. and Stat. Comput., vol. 13, no. 6, pp. 1488–1501.

[16] Shampine L.F., Thompson S. Solving DDEs in MATLAB // Applied Numerical Mathematics, 2001, vol. 37, pp. 441–458.

[17] Shampine L.F. Dissipative approximations to neutral delay differential equations // Applied Mathematics and Computation, 2008, vol. 203, pp. 641–648.

[18] Shampine L.F. Solving ODEs and DDEs with Residual Control // Applied Numerical Mathematics, 2005, vol. 52, pp. 113–127.

[19] Хайрер Э., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Том 2. Жёсткие и дифференциально-алгебраические задачи — М.: Мир, 1999. 685 с.

[20] Хайрер Э., Нерсётт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Том 1. Нежёсткие задачи — М.: Мир, 1990. 512 с.

[21] Paul C.A.H. A test set of functional differential equations — NA Report No. 243, Dept. of Mathematics, Univ. of Manchester, 1994.

Приложение

```
function sol = moiseev_ddensd (ddefun, delays, ddelays, history, ↩
dhistory, tspan, options)

    if nargin < 7       %проверка кол-ва ↩
    входных аргументов
        options = [];
        if nargin < 6
            error('Not Enough Inputs');
        end
    end

    if ~isa(ddefun,'function_handle')    %проверка ↩
    заданной функции ddefun
        error('ddefun Not Function Handle');
    end
    if ~isa(delays,'function_handle')    %проверка ↩
    заданной функции delays
        error('delays Not Function Handle');
    end
    if ~isa(ddelays,'function_handle')    %проверка ↩
    заданной функции ddelay
        error('history Not Function Handle');
    end
    if ~isa(history,'function_handle')    %проверка ↩
    заданной функции history
        error('history Not Function Handle');
    end
    if ~isa(dhistory,'function_handle')    %проверка ↩
    заданной функции dhistory
        error('dhistory Not Function Handle');
    end
    if ~isnumeric(eps)   %проверка типа ↩
    заданной точности
        error('eps Invalid Type');
    end
    if ~isnumeric(tspan)   %проверка типа ↩
    заданного интервала
        error('Span Invalid Type');
    end
```

```
    t_start = tspan(1);
    t_final = tspan(end);

    if t_final <= t_start    %проверка заданного ↵
    интервала
        error('The Span Is Wrong');
    end

    htry = ddeget(options,'InitialStep',[],'fast');
    if ~isempty(htry) && (htry <= 0)
        error(message('Invalid initial step'));
    end

    rtol = ddeget(options,'RelTol',1e-3,'fast');
    if ~isscalar(rtol) || (rtol <= 0)
        error(message('MATLAB:ddesd:OptRelTolNotPosScalar'));
    end
    atol = ddeget(options,'AbsTol',1e-6,'fast');
    if any(atol <= 0)
        error(message('MATLAB:ddesd:OptAbsTolNotPos'));
    end

    normcontrol = strcmp(ddeget(options,'NormControl','off','fast')↵
    ,'on');

    if normcontrol
      if ~isscalar(atol)
        error(message('MATLAB:ddesd:NonScalarAbsTolNormControl'));
      end
      normy = norm(history(t_start));
    else
      if ~isscalar(atol) && (numel(atol) ~= neq)
        error(message('MATLAB:ddesd:AbsTolSize', funstring( ddefun
↵
        ), neq));
      end
      atol = atol(:);
    end
    threshold = atol / rtol;

    hmax = min(t_final - t_start, ddeget(options,'MaxStep',0.1*(↵
    t_final - t_start),'fast'));
```

```
    if hmax <= 0
      error(message('Invalid OptMaxStep'));
    end

    if normcontrol
      if ~isscalar(atol)
        error(message('MATLAB:ddesd:NonScalarAbsTolNormControl'));
      end
      normy = norm(history(t_start));
    else
      if ~isscalar(atol) && (numel(atol) ~= neq)
        error(message('MATLAB:ddesd:AbsTolSize', funstring( ddefun
↪
        ), neq));
      end
      atol = atol(:);
    end
    threshold = atol / rtol;
    threshold = cast(threshold,'double');

    %Инициализация (1 шаг)
    nsteps = 0;
    nfevals = 0;
    y_start = history(t_start);
    t = t_start;
    y = y_start;
    dy = ddefun(t_start, y_start, delay_vals(t_start, y_start), ↪
    ddelay_vals(t_start, y_start));
    k_history = [0 0 0 0 0 dy];
    sm = [(1/2 - sqrt(3)/6), (1/2 + sqrt(3)/6)];

    nfevals = nfevals + 1;
    b = coef_b_vals(1);     % значения ↪
    коэффициентов b
    t_curr = t_start;

    pow = 1/4;
    hmin = 16*eps(t_start);
    if isempty(htry)
      % Compute an initial step size h using y'(t).
      h = min(hmax, t_final - t_start);
      if normcontrol
```

```
        rh = (norm(dy) / max(normy,threshold)) / (0.8 * rtol^pow);
      else
        rh = norm(dy ./ max(abs(y),threshold),inf) / (0.8 * ↵
        rtol^pow);
      end
      if h * rh > 1
        h = 1 / rh;
      end
      h = max(h, hmin);
    else
      h = min(hmax, max(hmin, htry));
    end

  while true

      hmin = 16*eps(t(end));
      h = min(hmax, max(hmin, h));    % couldn't limit h until
↵
      new hmin

      %растояние до конца отрезка
      distance = t_final - t_curr;

      %подгоняем последний шаг
      if h > distance
          h = distance;
      end


      t0 = t(end);
      y0 = y(end);

      nofailed = true;

      while true
          t_curr = t0 + h;

          k(1) = k_history(end);        %свойство FSAL
          y_temp = y0 + h*(1/6)*k(1);
          k(2) = ddefun(t0 + (1/6)*h,     y_temp, delay_vals(t0
+↵
          (1/6)*h, y_temp), ddelay_vals(t0 + (1/6)*h, y_temp));
```

```
            y_temp = y0 + h*((44/1369)*k(1) + (363/1369)*k(2));
            k(3) = ddefun(t0 + (11/37)*h,   y_temp, delay_vals(t0
+)
            (11/37)*h, y_temp), ddelay_vals(t0 + (11/37)*h, y_temp)
            );
            y_temp = y0 + h*((3388/4913)*k(1) - (8349/4913)*k(2) +
)
            (8140/4913)*k(3));
            k(4) = ddefun(t0 + (11/17)*h,   y_temp, delay_vals(t0
+)
            (11/17)*h, y_temp), ddelay_vals(t0 + (11/17)*h, y_temp)
            );
            y_temp = y0 + h*(-(36764/408375)*k(1) + (767/1125)*k(2)
            - (32708/136125)*k(3) +(210392/408375)*k(4));
            k(5) = ddefun(t0 + (13/15)*h,   y_temp, delay_vals(t0
+)
            (13/15)*h, y_temp), ddelay_vals(t0 + (13/15)*h, y_temp)
            );
            y_temp = y0 + h*((1697/18876)*k(1) + (0)*k(2) + (
            50653/116160)*k(3) +(299693/1626240)*k(4) + (
            3375/11648)*k(5));
            k(6) = ddefun(t0 + (1)*h,       y_temp, delay_vals(t0
+)
            (1)*h, y_temp), ddelay_vals(t0 + (1)*h, y_temp));
            y_curr  = y0 + h*sum(b.*k);

            nfevals = nfevals + 5;
            nsteps = nsteps + 1;

            err = 0;
            if normcontrol
              normynew = norm(y_curr);
              wt = max(max(normy,normynew),threshold);
            else
              wt = max(max(abs(y0),abs(y_curr)),threshold);
            end
            for i = 1:2
              xm = t0 + h*sm(i);
              [ym,ypm] = ntrp3h(xm,t0,y0,t_curr,y_curr,dy(end),k(6)
              );
              res = ypm - ddefun(xm,ym,delay_vals(xm, ym),
              ddelay_vals(xm, ym));
```

```
      nfevals = nfevals + 1;
      if normcontrol
        err = 2.1342*max(err, h*norm(res) / wt);
      else
        err = 2.1342*max(err, h*norm(res ./ wt,inf));
      end
      if (err > rtol) && ~nofailed
        break;
      end
    end

    if err > rtol
        if nofailed
          h = h * max(cast(0.5,'double'), 0.8*(rtol/err)^
          pow);    % "Optimal" reduction
        else
          h = h * 0.5;                                  % ^
          Potential discontinuity
        end
        h = max(hmin, h);
        nofailed = false;
    else      % Successful step
        break;
    end


end

%Сохраняем найденные значения
t = [t, t_curr];
y = [y, y_curr];
dy = [dy, k(6)];
k_history = [k_history; k];

%выход из цикла
if abs(t_final - t_curr) < 1e-07
    break;
end

if nofailed
    % Require that 0.8 <= hnew/h <= 2.
    h = h / max(cast(0.5,'double'),1.25*(err/rtol)^pow);
```

```
        h = min(max(hmin,h),hmax);
    end
end

sol.x = t;
sol.y = y;
sol.dy = dy;
sol.stats.nfevals = nfevals;
sol.stats.nsteps = nsteps;
sol.k_history = k_history;

function delay_val = delay_vals(t_curr, y_curr)

    delay = delays(t_curr, y_curr);
    for i=1:length(delay)           %для каждого ↵
    запаздывания находим значение
        if delay(i) <= t_start
            delay_val(i) = history (delay(i));
        else
            it = find (t <= delay(i)); % ищем ↵
            промежуток в который ↵
            попадает запаздание
            indx = it(end);
            if ~(indx == size(t))
                t_prev = t(indx);
                t_next = t(indx + 1);
                y_prev = y(indx);
                theta = (delay(i) - t_prev)/(t_next - t_prev);
                b_temp = coef_b_vals(theta);
                k_temp = k_history (it(end) +1, :);
                delay_val(i)  = y_prev + (t_next - t_prev)*sum(↵
                b_temp.*k_temp);
            else
               % проблема в оверлепинге
               delay_val(i) = history(delay(i));
            end
        end
    end
end

function ddelay_val = ddelay_vals(t_curr, y_curr)
```

```
            ddelay = ddelays(t_curr, y_curr);
            for i=1:length(ddelay)            %для каждого ⤶
            запаздывания находим значение
                if ddelay(i) <= t_start
                    ddelay_val(i) = dhistory (ddelay(i));  % ⤶
                else
                    it = find (t <= ddelay(i)); % ищем ⤶
                    промежуток в который ⤶
                    попадает запаздание
                    indx = it(end);
                    if ~(indx == size(t))
                        t_prev = t(indx);
                        t_next = t(indx + 1);
                        theta = (ddelay(i) - t_prev)/(t_next - t_prev);
                        db_temp = coef_db_vals(theta);
                        k_temp = k_history (it(end) + 1, :);
                        ddelay_val(i)  = sum(db_temp.*k_temp);
                    else
                        % проблема в оверлепинге
                        ddelay_val(i) = dhistory (ddelay(i));
                    end
                end
            end
        end
    end

    function coef_b = coef_b_vals(theta)   % функция ⤶
    возращающая коэфициентов b
        coef_b(1) = -(866577/824252)*theta^4 + (1806901/618189)⤶
        *theta^3 - (104217/37466)*theta^2 + theta;
        coef_b(2) = 0;
        coef_b(3) = (12308679/5072320)*theta^4 - (2178079/380424)⤶
        *theta^3 + (861101/230560)*theta^2;
        coef_b(4) = -(7816583/10144640)*theta^4 + (6244423/5325936)⤶
        *theta^3 - (63869/293440)*theta^2;
        coef_b(5) = -(624375/217984)*theta^4 + (982125/190736)*theta^3
    ⤶
        - (1522125/762944)*theta^2;
        coef_b(6) = (296/131)*theta^4 - (461/131)*theta^3 + (165/131)⤶
        *theta^2;
    end
```

```
function coef_db = coef_db_vals(theta)    % функция ⤶
возращающая коэфициенты b взятые по ⤶
производной
    coef_db(1) = -4*(866577/824252)*theta^3 + 3*(1806901/618189)⤶
    *theta^2 - 2*(104217/37466)*theta + 1;
    coef_db(2) = 0;
    coef_db(3) = 4*(12308679/5072320)*theta^3 - 3*(2178079/380424)⤶
    *theta^2 + 2*(861101/230560)*theta;
    coef_db(4) = -4*(7816583/10144640)*theta^3 + 3*(⤶
    6244423/5325936)*theta^2 - 2*(63869/293440)*theta;
    coef_db(5) = -4*(624375/217984)*theta^3 + 3*(982125/190736)⤶
    *theta^2 - 2*(1522125/762944)*theta;
    coef_db(6) = 4*(296/131)*theta^3 - 3*(461/131)*theta^2 + 2*(⤶
    165/131)*theta;
end
```