Моделирование химических и экологических процессов

УДК 51-7:502.52 DOI: 10.25514/CHS.2025.2.29002

Так ли жестки “жесткие системы” дифференциальных уравнений?

С. О. Травин

Федеральное государственное бюджетное учреждение науки Федеральный исследовательский центр химической физики им. Н.Н. Семенова Российской академии наук, Москва, Россия, e-mail: travinso@yandex.ru

Поступила в редакцию: 13.10.2025 г.; после доработки: 28.10.2025 г.; принята в печать: 17.11.2025 г.

Аннотация – На примере известной жесткой схемы реакций Робертсона показано практическое применение различных методов интегрирования прямой задачи химической кинетики. Проведено сравнение одношаговых и многошаговых методов, явных и неявных схем решения и разработан оригинальный подход, сочетающий применение метода Адамса на начальной стадии развития реакции с переходом к системе дифференциально-алгебраических уравнений, решаемых методом Эйлера после достижения стационарного режима. Достигнуто рекордно короткое общее время численного решения задачи на интервале времен свыше 20 порядков. Предложенный метод превосходит традиционные алгоритмы по быстродействию не менее, чем в сто раз.

Ключевые слова: разностные схемы, метод Эйлера, метод Адамса, метод Розенброка

Simulation of chemical and ecological processes

UDC 51-7:502.52 DOI: 10.25514/CHS.2025.2.29002

Are the “stiff systems” of differential equations so stiff?

Sergey O. Travin

N.N. Semenov Federal Research Center for Chemical Physics, Russian Academy of Sciences, Moscow, Russia, e-mail: travinso@yandex.ru

Received: October 13, 2025; Revised: October 28, 2025; Accepted: November 17, 2025

Abstract – Using the example of a well-known stiff Robertson reaction scheme, the practical application of various methods for integrating the direct problem of chemical kinetics is shown. A comparison of one-step and multi-step methods, explicit and implicit solution schemes was made and an original approach was developed that combines the application of the Adams method at the initial stage of reaction development with the transition to a system of differential-algebraic equations solved by the Euler method after achieving a stationary mode. A record short total computer time has been reached for the numerical solution of the problem over a time interval of more than 20 orders of magnitude. The proposed method is at least one hundred times faster than traditional algorithms.

Keywords: difference schemes, Euler method, Adams method, Rosenbrock method

ВВЕДЕНИЕ

Закономерности химической кинетики имеют первостепенное значение при решении задач экологической химии, фармакокинетики, токси-кинетики и ряда других важнейших направлений, связанных с поведением и/или распространением опасных веществ в окружающей среде и организме людей и животных. Причем в большинстве случаев процессы накопления веществ, претерпевающих биотрансформации, развиваются во времени по законам , существенно более сложным, чем реакции первого порядка. Но экспериментальная проверка полученных уравнений довольно сложна, поэтому использование кинетического метода анализа кривых зависимости концентрация (доза) – время, которая позволяет определять и прогнозировать основные параметры токсичности и опасности химических соединений становятся приоритетными.

Решение кинетических задач в химико-экологическом моделировании – это процесс построения математических моделей, которые описывают динамику процессов в сложных химико-экологических системах. Такие задачи могут быть связаны с изучением изменений во времени концентраций компонентов системы, пространственного перераспределения компонентов или других явлений.

В математическую модель закладываются, в том числе, биологические представления и гипотезы о кинетических свойствах процессов (скоростях роста, размножения, гибели, интенсивностях взаимодействия).

Детерминированное моделирование чаще всего основывается на системе дифференциальных уравнений с заданными начальными условиями, что известно как задача Коши. Например, система уравнений Лотки-Вольтерры используется для моделирования колебательных процессов в сообществах типа «хищник–жертва» с наличием саморазвивающихся (автокаталитических) стадий.

По-видимому, основной сложностью кинетического моделирования, как выяснилось в ходе исторического развития этого направления, стала работа с так называемыми жесткими системами (строгого определения этого названия поныне не существует, но необходимые пояснения будут даны в тексте этой публикации.

В химической кинетике центральной математической проблемой является решение так называемой прямой задачи, т.е. интегрирование системы обыкновенных дифференциальных уравнений с заданными начальными условиями (задача Коши).

$$\dot{y} \equiv \frac{dy}{dt} = f(t, y), \qquad y(t_0) = y_0, \qquad t_0 \le t \le t_k \tag{1}$$

Отметим, что (1) представляет собой не единичное уравнение, а систему уравнений, в которой y представляет собой вектор концентраций, размерности,

соответствующей количеству рассматриваемых компонентов химической системы. Исторически первым численным методом интегрирования систем обыкновенных дифференциальных уравнений, несомненно, является разностная схема:

$$y_{n+1} = y_n + (t_{n+1} - t_n) \cdot f(t_n, y_n)\,, \qquad f(t,y) = \frac{dy}{dx},\ y_{|t=t_0} = y_0 \qquad (2)$$

Предложенная Эйлером в 1768 году в работе «Интегральное исчисление» схема является одношаговым методом первого порядка точности. Он основан на аппроксимации интегральной кривой кусочно-линейной функцией, т.е. теоретически простейшим способом нахождения численного решения. Формула (2) представляет собой так называемую явную схему Эйлера, в которой для вычисления значения функции y_{n+1}на правом конце отрезка (принято, что время течет по оси абсцисс слева направо) используется значение скорости $f(t_n, y_n)$ для уже известного вектора y_n, посчитанного на предыдущем шаге.

Схема Эйлера и по сей день остается актуальной и востребованной, за время ее существования появились сотни усовершенствований, включая многошаговые методы (до 5 шагов) и методы более высокого порядка точности. Так, например, алгоритм и программа численного интегрирования методом Эверхарта были разработаны до 27 порядка, однако использование этих алгоритмов свыше 19-го порядка не приводило к повышению точности вычислений. В монографии [2] рассмотрены модифицированные алгоритмы Эверхарта до 31 порядка включительно, повышающие эффективность при увеличении порядка метода.

Однако использование алгоритмов, эксплуатирующих всё возрастающую вычислительную мощность современных компьютеров, представляется тупиковым направлением развития. Более перспективным, по-видимому, является подход, основанный на применении нейронных сетей и искусственного интеллекта [3].

Проводя ретроспективный анализ развития численных методов, можно выделить несколько знаковых событий [4, 5]. Серьезное усовершенствование схемы Эйлера, породившее целое семейства многошаговых методов, было сделано Адамсом в 1855 г. Идея Адамса состояла в том, чтобы для вычисления очередного значения искомого решения использовать не одно, а несколько значений, уже вычисленных в предыдущих точках. К достоинствам этих методов следует отнести то, что в них можно изменять и шаг интегрирования и порядок метода. Доказано, что для многошаговых численных методов Адамса при решении задачи Коши до 12 порядка область устойчивости уменьшается. Наибольшее распространение получили методы Адамса пятого порядка точности, использующие вычисление функции и ее производной в пяти точках. На практике широко используются как явные, так и неявные варианты методов Адамса: первые известны как методы Адамса-Бэшфорта, а вторые – как методы Адамса-Мултона.

Еще через полвека похожие идеи получили дальнейшее развитие в работах немецких математиков К. Рунге и М. В. Кутты. Для улучшения точности разностной схемы они воспользовались разложением вычисляемой функции в ряд Тейлора в окрестности точки, от которой делается очередной шаг. На первый взгляд, производные любого порядка можно непосредственно вычислить по формулам последовательного дифференцирования уравнения (1). Однако получаемые при этом формулы даже в операторной форме оказываются чрезмерно громоздкими, что снижает их практическую ценность. В связи с этим К. Рунге предложил, а В. Кутта развил идею метода вычисления высших производных в виде линейных комбинаций первых производных, вычисленных в промежуточных точках шага. Наиболее часто используется (и реализован в таких математических пакетах, как MathCAD, Maxima) классический метод Рунге - Кутты, имеющий четвёртый порядок точности.

При одинаковой точности для многошаговых методов Адамса на одном шаге интегрирования требуется меньше вычислений правых частей дифференциальных уравнений, чем в методах Рунге-Кутты. Практика также показала хроническую слабость методов Рунге-Кутты применительно к т.н. жестким системам ОДУ. Жесткой системой обыкновенных дифференциальных уравнений принято называть (нестрого) такие системы, численное решение которых может оказаться неудовлетворительным из-за резкого увеличения числа вычислений (при малом шаге интегрирования) или из-за резкого возрастания погрешности (так называемого, взрыва погрешности). Обычно это происходит, если различные компоненты системы имеют резко отличающиеся (на много порядков) характеристические времена релаксации (конфликт быстрой и медленной подсистем).

При анализе разностных схем для решения жестких задач, весьма частым предложением для анализа возможностей метода является установление собственных значений якобиана СОДУ. С точки зрения формальной математики такие рекомендации являются безусловно верными. При этом молчаливо предполагается, что для типового пользователя математических пакетов, применяемых в химической кинетике, процедура вычисления собственных значений является хорошо знакомой, что далеко не так. Еще менее понятно, как поступать, если вдруг окажется, что у собственных значений имеется мнимая часть.

Тем не менее ежегодно публикуются даже не десятки, а сотни работ, посвященных улучшению эффективности и устойчивости разностных схем, с помощью процедур анализа якобиана, оставляя при этом пользователей ЭВМ, решающих задачи, возникающие из инженерной практики, практически «безоружными».

В настоящей работе мы предлагаем весьма простые рецепты выбора вычислительных процедур, и приводим способы сравнения их эффективности.

Чем сложны «жесткие» системы?

Жесткие системы уравнений уже не первый десяток лет вызывают особый интерес, т.к. в химической кинетике очень часто встречаются

механизмы, в которых реакционная способность различных компонентов, а значит и их квазистационарная или квазиравновесная концентрация могут отличаться на несколько порядков – до десяти и более.

Существует множество определений жесткости системы, нам удобной представляется следующая. Прежде всего, надо отметить, что сами по себе те или иные уравнения не являются жесткими, жесткой может быть конкретная задача Коши, причем в определенных областях, размеры которых зависят от начальных значений и допустимой погрешности» [5]. И в этом смысле все сложности построения алгоритмов интегрирования таких систем сводятся не столько к выбору расчетной схемы, сколько к выбору оптимального шага разностной схемы, достаточно малого, чтобы обеспечить устойчивость и необходимую точность решения задачи, но при этом не превращающего вычислительную процедуру в томительное ожидание выполнений миллиардов внутренних циклов.

В монографии [6] отмечено, что при построении численных методов интегрирования жестких систем ОДУ основные тенденции связаны с расширением возможностей для решения систем все более высокой размерности. При этом, несмотря на рост быстродействия ЭВМ, сложность задач, встречающихся в практике, опережает развитие вычислительной техники, что приводит к только возрастающим требованиям к вычислительным алгоритмам.

Развитие численных методов показало, что универсального алгоритма, пригодного для решения любого класса жестких задач, создать не удалось и, по-видимому, его просто не существует. В каждом конкретном случае пользователю предстоит выбирать компромисс между потерей времени на вычисления и потерей точности и/или устойчивости решения. Основная проблема выбора состоит во взвешивании преимуществ и недостатков явных и неявных разностных схем.

Явные схемы отличаются очевидной простотой, не требуют ни решения систем уравнений, ни итерационных процедур, но на этом их «преимущества» заканчиваются. Единственным средством борьбы с жесткостью системы для них оказывается все более мелкое дробление шага, что в ряде случаев остается приемлемым, учитывая вычислительную дешевизну каждого шага.

Напротив, при реализации L-устойчивых неявных численных схем на каждом шаге решается линейная система алгебраических уравнений, например, с применением LU-разложения некоторой матрицы, размерность которой совпадает с размерностью вектора решения. При большой размерности исходной задачи общие вычислительные затраты фактически полностью определяются временем декомпозиции данной матрицы. Но в итоге количество элементарных шагов в таких схемах может оказаться на порядки меньше, чем в случае явных схем.

Дополнительные проблемы возникают в случае т.н. гибридных (дискретно-непрерывных) жестких систем, с разрывами функций правой части системы ОДУ. Применение многошаговых методов для решения таких задач едва ли возможно, поскольку в точках смены режима происходит потеря

исторической информации. Даже в случае использования одношаговых методов необходимы дополнительные ограничения на выбор величины шага в зависимости от поведения событийной функции.

1. Выбор модели и тактика построения сольвера

Надо заметить, что изобилие публикаций, посвященных решению прямой задачи химической кинетики сослужило дурную службу и затруднило возможность нахождения реально работающих рекомендаций. Преобладающее число литературных источников содержат в себе лишь общий набор формул, исключающий возможность имплементации в кодах. Так в поисковой системе Google Scholar запрос «direct problem of chemical kinetics», выдает свыше четырех миллионов ссылок, из которых для программной реализации пригодны лишь единицы.

Для тестирования возможностей и выявления проблемных мест при выборе вычислительного алгоритма была выбрана модель химических реакций Робертсона, которая является одним из первых и самых популярных примеров «жесткой» системы обыкновенных дифференциальных уравнений; она впервые опубликована в 1966 году [7] и в химической нотации имеет вид:

$$A \overset{k_1}{\rightarrow} B \qquad k_1 = 0.04, \qquad W_1 = k_1[A] \tag{3}$$

$$B + C \overset{k_2}{\rightarrow} A + C \qquad k_2 = 1 \cdot 10^4, \quad W_2 = k_2[B][C] \tag{4}$$

$$B + B \overset{k_3}{\rightarrow} C + B \qquad k_3 = 3 \cdot 10^7, \quad W_3 = k_3[B][B] \tag{5}$$

Модель Робертсона, описывается дифференциальными уравнениями:

$$\dot{y}_1 = -k_1 y_1 + k_2 y_2 y_3, \qquad y_1 \equiv [A] \tag{6}$$

$$\dot{y}_2 = \quad k_1 y_1 - k_2 y_2 y_3 - \quad k_3 y_2 y_2, \qquad y_2 \equiv [B] \tag{7}$$

$$\dot{y}_3 = \quad k_3 y_2 y_2 = k_3 y_2^2, \qquad y_3 \equiv [C] \tag{8}$$

Сам Робертсон отмечал: «Когда уравнения представляют поведение системы, содержащей ряд быстрых и медленных реакций, прямое интегрирование этих уравнений становится затруднительным».

Эта задача - самая жесткая из известных (мера жесткости $M_ж = 10^{15}$). Трудность ее решения в постановке Робертсона обусловлена не только сосуществованием быстрых и медленных реакций, но также и большим интервалом интегрирования (от 0 до 10^{11} с), на протяжении которого размер шага увеличивается более чем на 20 порядков. Большинство стандартных сольверов, особенно основанных на явном методе вычислений, оказались малопригодными для решения этой задачи.

Очевидно, напрашивается применение неявного метода Розенброка, однако, такой подход требует вычисления и инверсии Якобиана на каждом шаге и, кроме того, многократного вычисления правой части f в промежуточных точках, которые при переходе к новой точке повторно не используются. В качестве альтернативы с высокой производительностью

можно рассматривать многошаговые методы Адамса, но для них неизбежен сравнительно медленный «разгонный» участок для накопления необходимого числа шагов, определяемого порядком метода. В итоге мы пришли к композитному решению, в котором 5-шаговому методу Адамса предшествуют четыре шага по методу Розенброка.

2. Алгоритм Розенброка

В работе [8] разработан L-устойчивый трехстадийный метод Розенброка третьего порядка точности для решения жестких задач. Для него построено неравенство контроля точности вычислений, основанное на оценке аналога глобальной ошибки, которая осуществляется с привлечением ранее вычисленных стадий. Это позволяет выбирать величину шага интегрирования фактически без увеличения вычислительных затрат. Применительно к тестовой задаче мы убедились, применение этого метода, даже в упрощенном варианте не влечет неприятных последствий.

Так на каждом n-ом шаге интегрирования выполняется следующая последовательность действий:

Процедура 1. Вычисляется матрица Якоби $f_n' = \partial f(y_n)/\partial y$.

Процедура 2. Формируется матрица $D_n = E + ahf_n'$, где E – единичная матрица, h – шаг интегрирования, a = 0,435866521508459 – специально вычисленный для этого метода числовой коэффициент.

Процедура 3. Вычисляются стадии $k_1,\ k_2,\ k_3$ по формулам, приведенным в [8].

Процедура 4. Вычисляется оценка ошибки на n-м шаге $\varepsilon_n(1)$.

Процедуры 5–10. Если ошибка превышает заранее заданный порог точности осуществляется дробление шага пополам и возврат к пересчету по процедуре 3. Если же ошибка с трехкратным запасом не достигает установленного порога, осуществляем удвоение шага. Для диапазона ошибки $porog/3 \le \varepsilon \le porog$ шаг остается прежним.

Процедура 11 Переход к следующему шагу интегрирования.

Указанный набор процедур был воплощен в коде в среде VBA-Excel и подтвердил заявленную в [8] устойчивость. При этом следует отметить, что по быстродействию он примерно в 30 раз уступает алгоритму, предоставленному нам Е.А. Новиковым примерно 30 лет назад, хотя для первых четырех шагов такое замедление роли не играет.

3. Многошаговый алгоритм Адамса

Подробно метод Адамса описан в [9]. Рекомендуемым подходом является использование сочетания «предиктор – корректор». На первой стадии используется явный метод вычисления предиктора, т.е. получение предварительного значения вектора $y_{n+1}^* \equiv y^*(t+h)$ на n+1 шаге исходя из уже вычисленных значений на предыдущих пяти шагах по формуле Адамса-Бэшфорта:

$$y_{n+1}^* = y(t) + \frac{h}{720}\left[1901f\big(y(t)\big) - 2774f(y(t-h) + 2616f(y(t-2h) - 1274f(y(t-3h) + 251f(y(t-4h)\right] \quad (9)$$

Следующим действием, является эффективная подстановка предиктора в качестве аргумента для вычисления скорректированного выражения y_{n+1} по формуле Адамса-Молтона:

$$y_{n+1} = y(t) + \frac{h}{720}\left[251f\left(y^*(t+h)\right) + 646f(y(t) - 264f(y(t-h) + 106f(y(t-2h) - 19f(y(t-3h)\right] \quad (10)$$

Оказалось, что описанный в [9] подход «предиктор-корректор» хотя и заработал с первого раза после составления кодов для VBA-Excel и показал высокие скоростные характеристики, но сразу же продемонстрировал склонность к неустойчивости (рис. 1). Именно выбор величины шага оказывает определяющее влияние не только на вычислительные затраты, но и на устойчивость алгоритма.

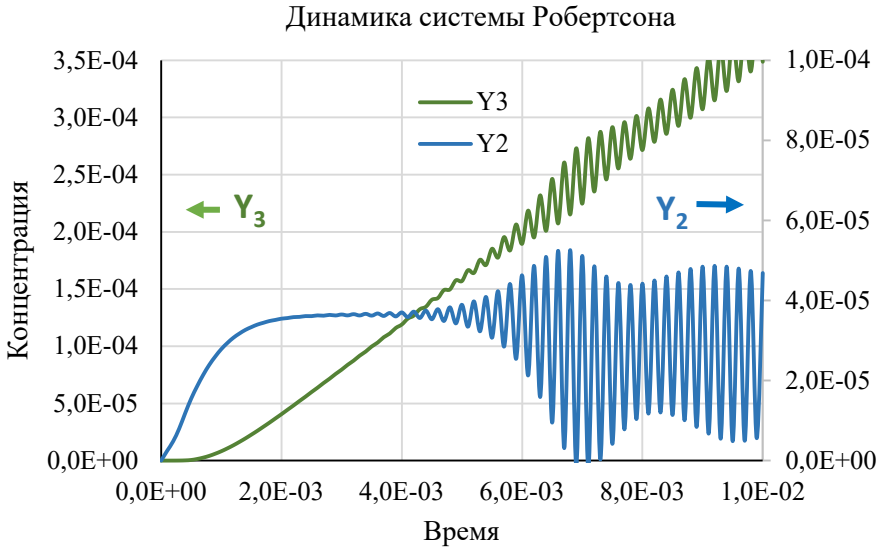

Рис. 1. Метод Адамса-Бэшворта-Молтона демонстрирует неустойчивость. Шаг интегрирования 0,0001 с. Затраченное машинное время 0,65 с.

Fig. 1. The Adams-Bashworth-Molton method demonstrates instability. The integration step is 0.0001 s. The machine time spent is 0.65 s.

4. Составной алгоритм

Перезапуск метода с более мелким шагом требует повторного накопления пяти (для выбранного порядка метода) стартовых точек, что влечет за собой дополнительные вычислительные затраты, но зато переводит алгоритм в устойчивый режим.

Эмпирически было подобрано пороговое значение различий между предиктором и корректором в (1–2) %. Повышение величины порога ведет к недопустимо грубым искажениям результата интегрирования, тогда как движение в сторону уменьшения приводит к резкому возрастанию затрат машинного времени. Убедившись в наличии «срыва» устойчивости при решении задачи Робертсона мы пошли по пути коррекции подхода «предиктор-корректор», а именно, в случае существенного различия между предикторным и корректорным значениями, осуществляли дробление шага пополам и возврат к значениям, полученным на предыдущем шаге (рис. 2).

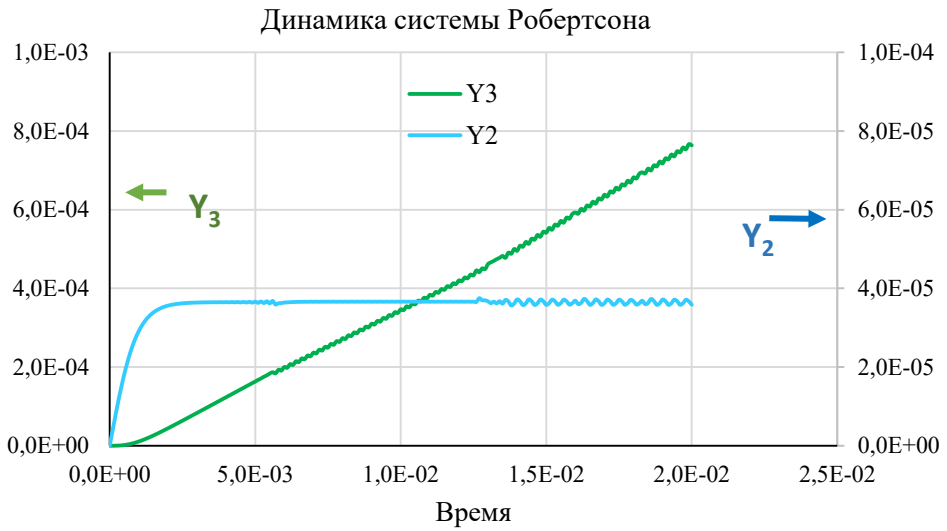

Рис. 2. Тот же метод с авторегулированием шага интегрирования. Затрачено 6,92 сек. машинного времени.

Fig.2. The same method with auto-regulation of the integration step. It took 6.92 seconds of machine time.

Начальный выбор в два раза более короткого шага ($5 \cdot 10^{-5}$ сек вместо $1 \cdot 10^{-4}$ сек) приводит к неожиданным последствиям. Помимо ожидаемого появления устойчивости и улучшения точности интегрирования, происходит весьма заметное сокращение общих вычислительных затрат.

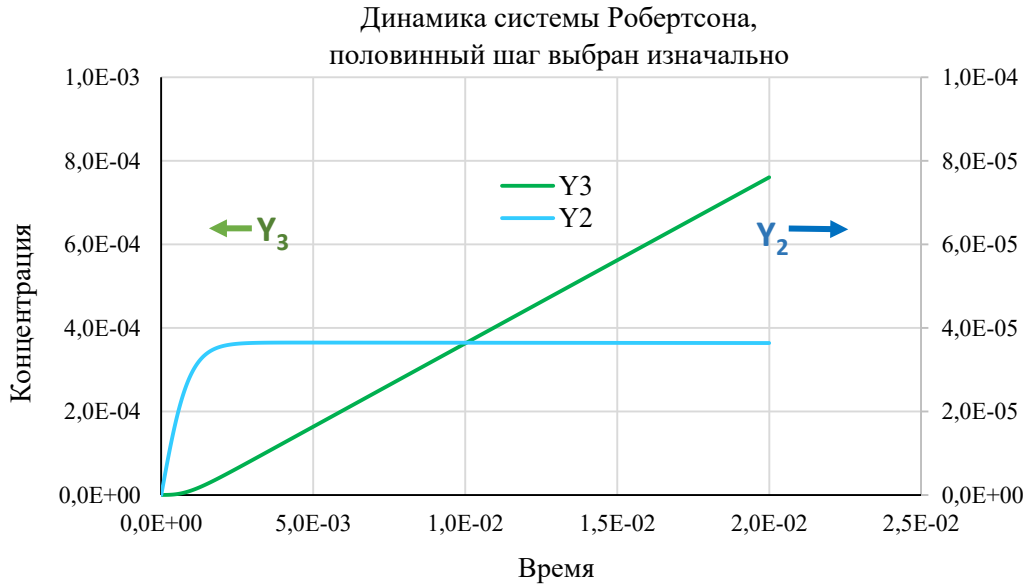

Рис. 3. Устойчивый и быстрый вариант расчета в случае выбора изначально малой стартовой величины регулируемого шага.

Fig. 3. Stable and fast calculation option in case of selection of initially small starting value of adjustable step.

Уменьшение шага в методе Адамса-Бэшворта-Молтона приводит не только к стабилизации расчетной процедуры, но и сокращает ее время.

Затрачено 2,24 сек. машинного времени. Шаг интегрирования 0,00005 сек., т.е. всего в два раза меньше, чем в предыдущих случаях.

Обратим внимание, что условно успешное применение композитного расчетного модуля (разгонные шаги по Розенброку, продолжение по Адамсу), графики которого приведены на рис. 1–3, относится к весьма ограниченному отрезку времени – до одной сотой секунды. В принципе, возможно экстенсивное применение того же способа с увеличением верхней границы запрашиваемого диапазона.

Проведенный вычислительный эксперимент показал, что алгоритм с наилучшей настройкой по шагу позволяет достичь границы t = 0.1 сек. приблизительно за 7 вычислительных секунд, а границу t = 1,0 сек. за 68 сек., и зависимость времени вычисления от заданной границы времени моделирования почти пропорциональная. Для временных диапазонов свыше сотни секунд время вычисления становится практически неприемлемым, и необходимо искать альтернативные варианты нахождения численного решения.

Естественной выглядит попытка увеличения шага после того, как пройдены особенности начального этапа. После прохождения системного времени t = 0,01 сек. шаг по времени может быть увеличен почти на два порядка (до $2 \cdot 10^{-3}$ секунды), однако уже через несколько шагов система начинает проявлять склонность к осцилляциям, и чтобы погасить возникающие колебания приходится снижать шаг до $5 \cdot 10^{-4}$ сек. В итоге все попытки увеличить шаг оказались неудачными (рис. 4).

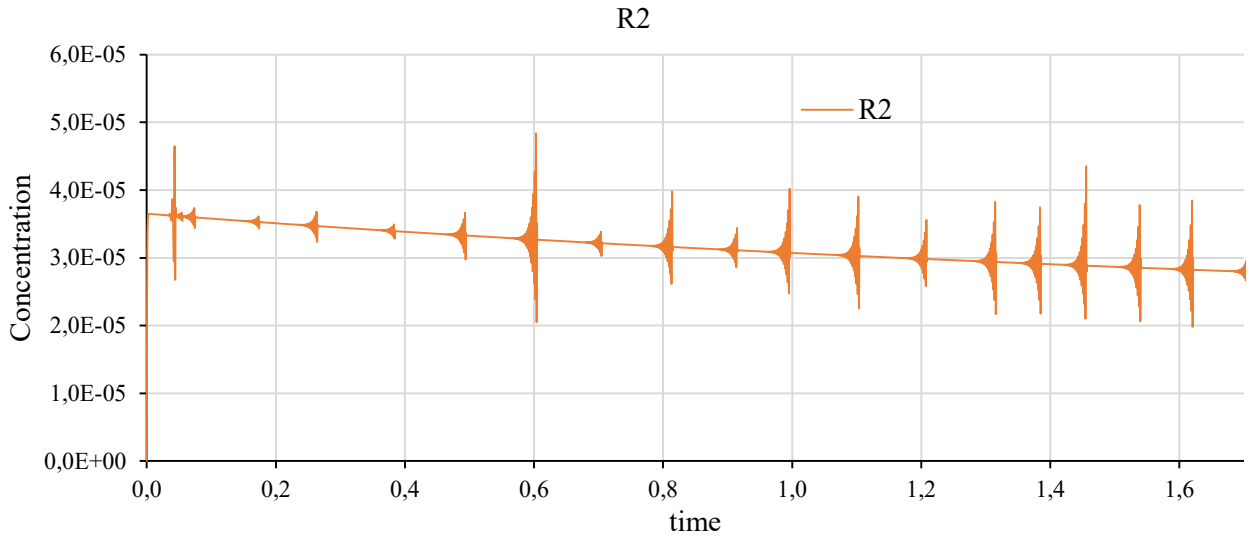

Рис. 4. Неустойчивость алгоритма Адамса-Бэшворта-Молтона при регулировании шага.

Fig. 4. Instability of the Adams-Bashworth-Molton algorithm in pitch control

В данном вычислительном эксперименте поводом для уменьшения шага вдвое служил переход результата счета в осциллирующий режим. Прохождение нескольких шагов (в пределах 10) интегрирования с укороченным шагом стабилизирует вычислительную процедуру, после чего возможен примерно стократный повтор разностной схемы с удвоенным размером шага. В любом случае попытка использовать шаг $h > 1 \cdot 10^{-3}$ сек. приводит к немедленному и

необратимому срыву вычислительной процедуры. Теоретически вместо визуального контроля за переходом в осцилляции можно поставить и программную проверку, но мы не стали этого делать ввиду бесперспективности дальнейшего увеличения шага сверх 1 мсек.

5. А можно ли проще?

Обратим внимание на то, что сложность вычислений в системе Робертсона связана с тем, что различные компоненты вектора решений $\boldsymbol{y(t)}$ различаются на много (иногда свыше десятка) порядков величины. Если подбирать шаг схемы Эйлера так, чтобы гладко вычислялся самый малочисленный компонент, то шаг оказывается слишком мелким для остальных, макроскопически значимых компонентов. В результате для того, чтобы изменение для них было сколько-нибудь значимым и заметным, необходимо сделать астрономически огромное число шагов.

Напротив, если шаг подбирать по обильно присутствующему компоненту (в системе Робертсона это R_1), то для изменения его концентрации, на 0,1% выглядит разумным шаг $h = 0{,}025$ сек. Однако, таких шагов расчетный алгоритм допускает не более двух; причем уже на втором шаге концентрация R_2 уходит в отрицательную область, а заодно нарушается закон сохранения вещества: сумма концентраций $R_1 + R_3$ превышает исходную $R_1 + R_3 > R_1(0) = 1$.

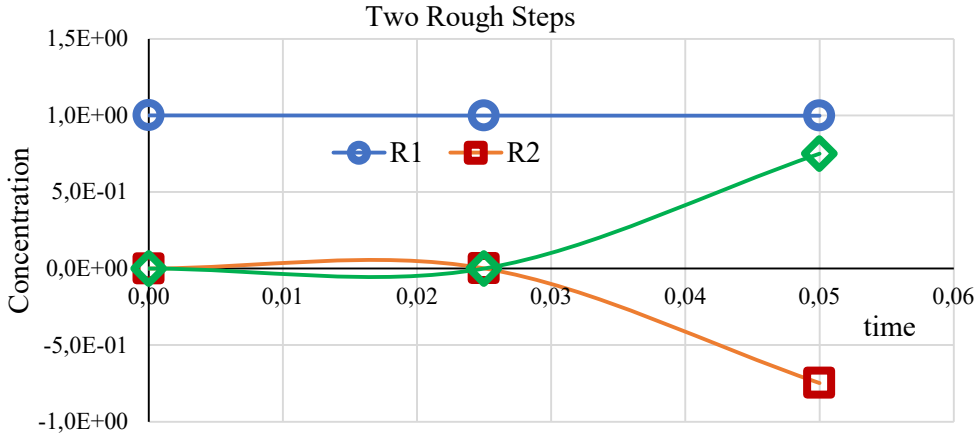

Рис. 5. Уже второй шаг схемы Эйлера, незаметный для вещества R_1, оказывается критическим для R_2 и R_3. Нет необходимости проверять, что последующие шаги окажутся только хуже. На седьмом шаге наступает переполнение (overflow) и компьютер отказывается выполнять дальнейшие расчеты.

Fig. 5. Though the second step of the Euler scheme, unimportant for the substance R_1, turns out to be critical for R_2 and R_3. There is no need to check that the next steps will only turn out worse. At the seventh step, an overflow occurs and the computer refuses to perform further calculations.

Эти досадные расчетные сложности возникают исключительно из-за того, что правильные (вычисленные по более устойчивой схеме) значения концентраций компонентов в начале развития реакции отличаются примерно на пять-семь порядков. Можно попытаться сделать преобразование переменных системы, которое устранит эту сложность.

Действие первое – логарифмирование

При переходе от самих концентраций к их логарифмам, ситуация с большим различием в порядках явно упрощается. Правда, для самой нулевой точки (начальные условия) некоторые концентрации имеют нулевое значение и логарифмированию не подлежат. Можно , например, принять, что в нулевой момент времени концентрации этих веществ очень маленькие, но уже не нулевые, скажем по 10^{-10} (в тех же единицах, в которых R_1 имеет единичную начальную концентрацию). Запас точности более чем достаточный.

Введем обозначения для замены переменных и посмотрим, как преобразуются правые части системы дифференциальных уравнений.

$$z_1(t) = \ln R_1(t) \qquad \frac{dz_1}{dt} = \frac{1}{R_1}\frac{dR_1}{dt} \tag{11}$$

$$z_2(t) = \ln R_2(t) \qquad \frac{dz_2}{dt} = \frac{1}{R_2}\frac{dR_2}{dt} \tag{12}$$

$$z_3(t) = \ln R_3(t) \qquad \frac{dz_3}{dt} = \frac{1}{R_3}\frac{dR_3}{dt} \tag{13}$$

$$\dot{z}_1 = -k_1 + k_2 \cdot \exp(z_2) \cdot \exp(z_3)/\exp(z_1) \tag{14}$$

$$\dot{z}_2 = \quad k_1 \cdot \exp(z_1)/\exp(z_2) - k_2 \cdot \exp(z_3) - \quad k_3 \cdot \exp(z_2) \tag{15}$$

$$\dot{z}_3 = \quad k_3 \cdot \exp(2z_2)\,/\exp(z_3) \tag{16}$$

Совершенно очевидно, и численный эксперимент подтверждает, что после логарифмирования происходит очень заметное «смягчение» ранее жесткой системы Робертсона. Тем не менее, некоторая жесткость все-таки сохраняется, поэтому простейшей разностной схеме Эйлера мы предпочли чуть более сложную, но заметно более устойчивую схему с элементами неявного подхода.

Возможны различные вариации сочетания явного и неявного методов Эйлера; существует целое семейство таких методов, параметризованных по α и задаваемых формулой:

$$y_{n+1} = y_n + h\left((1 - \tfrac{1}{2\alpha})f(t_n, y_n) + \tfrac{1}{2\alpha} f\big(t_n + \alpha h, y_n + \alpha h f(t_n, y_n)\big)\right) \tag{17}$$

В этом семействе α=1/2 дает метод средней точки, α=1 – метод Хойна (Karl Heun, 1886), и α=2/3 – это метод Ралстона.

Метод Эйлера-Хойна

Метод Эйлера позволяет получить приближение, сколь угодно близкое к точному решению. Однако точность повышается только линейно с уменьшением размера шага. Другими словами, для достижения аппроксимации, которая в 10 раз точнее, требуется в 10 раз больше шагов. Внеся небольшое изменение, можно получить метод, точность которого повышается квадратично, и так что увеличение числа шагов в 10 раз дает приближение, которое в 100 раз точнее. Идея улучшенного метод Эйлера, также называемого

метода Эйлера-Хойна, состоит в том, чтобы при интегрировании вместо прямоугольников использовать трапеции.

Единственная проблема заключается состоит в том, что значение y_{n+1}, которое мы хотим приблизить, появляется не только в левой, но и в правой части уравнения. Чтобы обойти эту «сложность», Карл Хойн применил метод Эйлера для аппроксимации $\hat{y}_{n+1}$, а затем использовал это значение в правой части уравнения для получения улучшенной аппроксимации. Эту разновидность разностной схемы мы и применили для интегрирования отлогарифмированной системы Робертсона.

Результаты моделирования по алгоритму Эйлера-Хойна представлены на рисунках 6 и 7. Быстродействие алгоритма даже превзошло ожидания. Алгоритм оказался примерно в 100 - 200 раз более эффективным, чем композитный алгоритм с разгонным участком по схеме Розенброка и продолжением в виде пятишагового метода Адамса в режиме «предиктор - корректор». Для сравнения: расчет временного интервала до 0,025 сек. с помощью композитного алгоритма потребовал свыше 2 сек машинного времени, тогда как для интервала до 0,05 сек. алгоритм Эйлера-Хойна уложился – менее, чем в две сотых секунды (0,02 сек.).

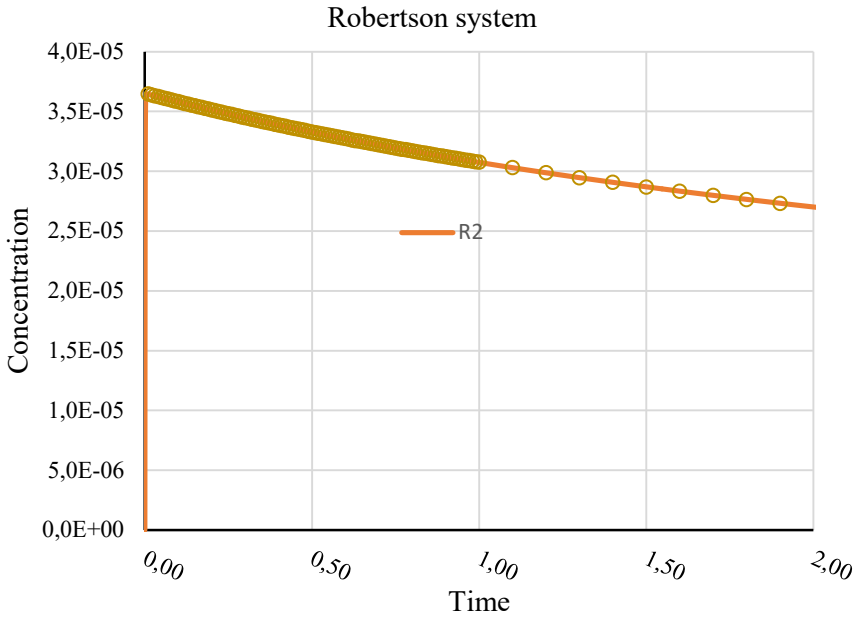

Рис. 6. Сплошной линией представлено моделирование временной динамики R2 по алгоритму Эйлера-Хойна. Просчитано 2500 шагов с саморегулированием. Затрачено 0,51 сек. машинного времени. Кружками представлено алгебраическое вычисление той же концентрации в квазистационарном приближении.

Fig. 6. The solid line represents the simulation of the time dynamics of R2 using the Euler-Hoyne algorithm. Calculated 2500 steps with self-regulation. It took 0.51 seconds of machine time. Circles represent an algebraic calculation of the same concentration in a quasi-stationary approximation.

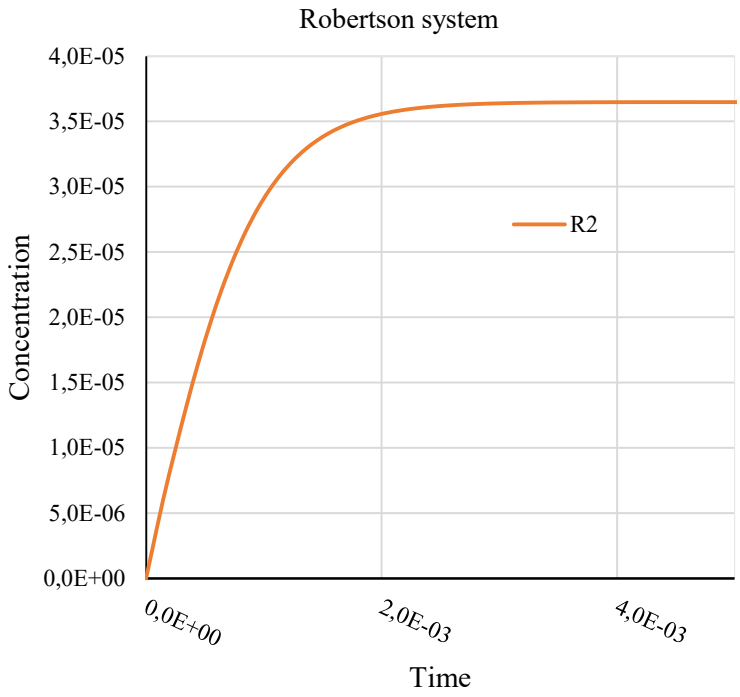

Рис.7. На самом деле достаточно промоделировать первые сто шагов – фактически до достижения наиболее вариабельным компонентом квазистационарной концентрации $R_2 \approx 3.65 \cdot 10^{-5}$, на что интегратор Эйлера-Хойна затрачивает 0,02 сек машинного времени.

Fig.7. In fact, it is enough to simulate the first hundred steps - in fact, until the most variable component reaches the quasi-stationary concentration R _ 2 ≈ 3.65 · 10 ^ (-5), which the Euler-Hoyne integrator spends 0.02 seconds of machine time.

Мы убедились в том, что схема относительно толерантна к выбору шага. Для достижения высокой точности счета первый шаг был сознательно выбран очень маленьким – 10^{-10} с, при этом мы убедились, что даже увеличение величины стартового шага на три порядка не приводит к ощутимому ухудшению точности.

Каждый следующий шаг, начиная со второго, увеличивался на полпорядка, по сравнению с предыдущим. Таким образом за двенадцать шагов величина инкремента по времени увеличивается от $1 \cdot 10^{-10}$ сек. до $1 \cdot 10^{-4}$ сек, а затем право варьировать шаг предоставлено самой программе. Если интегрирование проходит гладко, шаг можно удвоить. Если же у расчетной процедуры начинают проявляться склонности к осцилляциям, шаг необходимо уменьшить вдвое. В качестве критерия устойчивости поведения алгоритма мы выбрали величину относительного изменения скорости накопления (или расхода) R_2 – она не должна превышать некоего порогового значения, в качестве которого мы приняли 35% (этот выбор не критичен). Чтобы избежать излишних «дерганий» и уменьшить временные затраты на тестовые проверки, программа может изменять шаг в ту или иную сторону один раз в пять шагов. После этого до следующей проверки программа оставляет его неизменным.

Важно, что при последовательном удвоении шага мы ограничили его предельную величину значением $1{,}6 \cdot 10^{-3}$ сек. Логика программы и без нашего вмешательства в режиме авторегулирования шага успешно избегает

превышения порогового значения. Фактически мы ввели пороговое ограничение лишь для того, чтобы обезопасить себя от переполнения на шагах между проверками. Заметим, что Все использованные нами регулировочные параметры алгоритма носят сугубо рекомендательный характер и могут варьироваться в довольно широких пределах по желанию пользователя без заметного ущерба для устойчивости алгоритма, но, возможно, с некоторыми незначительными потерями по быстродействию.

Полный листинг программы интегрирования с регулированием шага приведен в приложении к настоящей работе. Мы не включили в него код вычисления функций скорости по формулам (14-16), но самостоятельное его написание не представляет никакой сложности.

Тем не менее, даже после логарифмирования (которое позволило очень быстро промоделировать начальный участок развития реакций в системе Робертсона), набор ОДЕ все-таки остается жестким. И чисто экстенсивное увеличение числа шагов интегрирования заведомо не приведет к успеху, тем более что наложенное ограничение по шагу по времени приводит к нелинейному возрастанию вычислительного времени. Если вернуться к первоначальному диапазону времен моделирования, затребованному Робертсоном ($0 \le t \le$ до 10^{11} сек.), даже шаг в миллион секунд не представляется избыточно большим, поскольку по-прежнему требуется очень большое количество шагов интегрирования. Тем самым, задача упрощения системы и укрупнения шага становится актуальной, как никогда ранее.

5.1. Действие второе – вместо дифференциальных уравнений алгебраические

Выход из положения подсказывает сама жесткость системы. Чем вызваны вычислительные проблемы укрупнения шага? Тем, что процедура может срываться в колебательный режим. Происходит это от того, что в выражение для скорости изменения R_2 входят члены разного знака:

$$\dot{y}_2 = \quad k_1 y_1 - k_2 y_2 y_3 - \quad k_3 y_2 y_2, \qquad y_2 \equiv [B] \tag{7'}$$

Анализ числовых значений членов уравнения (7), полученных по Эйлеру-Хойну, показал, что два члена из трех много больше, чем их алгебраическая сумма. Это правило выполняется повсюду, кроме нескольких первых шагов. Поэтому результирующее значение скорости может оказаться знакопеременным, поскольку в приблизительных вычислениях мы определяем малую величину, как разность двух больших, что, приводит к большой ошибке вычисления. Т.е, крупные члены в правой части (7) как бы уничтожают друг друга и результат (с точностью до ошибки аппроксимации) дают нулевой, и мы приходим к приближенному уравнению (18)

$$k_1 y_1 - k_2 y_2 y_3 - \quad k_3 y_2 y_2 \approx 0 \tag{18}$$

К такому же выводу приводит и хорошо известный в химической кинетике, но к настоящему времени почти забытый метод квазистационарных (вариант – квазиравновесных) концентраций. В первоначальной формулировке

Макса Боденштейна (1933) он выглядел примерно так: если реакция протекает в несколько стадий с образованием промежуточных продуктов, имеющих небольшие концентрации, то можно допустить, что концентрации стационарны, то есть не меняются со временем.

Необходимым и достаточным условием малости промежуточных скоростей является высокая реакционная способность промежуточных продуктов. Она приводит к тому, что эти высокоактивные вещества не накапливаются в системе, а проходят транзитом, а если малы сами концентрации, то обязаны быть малыми и их производные по времени.

Применённый нами подход может быть использован для численного моделирования не только химических систем, но гораздо более широкого класса задач. Именно из-за того, что различные компоненты вектора решения отличаются на много порядков и возникает жесткость, поэтому есть основания считать малочисленные компоненты активными и приближать их концентрации квазистационарными алгебраическими уравнениями.

Например, адиабатическое приближение в ядерной физике представляет собой разделение системы на тяжёлые и лёгкие частицы — ядра и электроны. Вследствие резкого различия в их массах и скоростях, можно считать, что для описания движения электронов можно считать тяжелые частицы (ядра) как бы неподвижными. Напротив, медленное движение тяжеловесных ядер происходит в усредненном поле, вычисляемом как среднее пространственное распределение электронов. Этот метод приближённого решения задач квантовой механики, в которых можно выделить быструю и медленную подсистемы. Поставленная задача решается по частям: сначала на первом этапе рассматривается движение быстрой подсистемы при фиксированных координатах медленной подсистемы, а затем учитывается движение последней.

Возвращаясь к уравнению (18), мы можем (и должны) дополнить его неравенствами:

$$y_2 \ll y_1; \qquad y_2 \ll y_3 \tag{19}$$

Их справедливость вытекает из высокой реакционной способности частицы B из системы Робертсона, поскольку для реакций с ее участием константы скорости заметно выше остальных; это же подтверждается и непосредственными численными расчетами. Отсюда следует:

$$y_1(t) + y_3(t) \approx y_1(0) = 1 \tag{20}$$

или

$$y_3(t) \approx 1 - y_1(t) \tag{21}$$

Подставляя это выражение в (18) получаем квадратное уравнение (алгебраическое, а не дифференциальное!) для y_2

$$k_1 y_1 - k_2 y_2 (1 - y_1) - \quad k_3 y_2^2 \approx 0 \tag{22}$$

или

$$y_2^2 + k_2/k_3 (1 - y_1) y_2 - k_1/k_3 y_1 \approx 0 \tag{22}$$

единственным положительным решением которого является

$$y_2 = \frac{k_2(y_1 - 1)}{2k_3} + \frac{1}{2}\sqrt{4y_1 \cdot k_1/k_3 + (k_2/k_3(1 - y_1))^2} \quad (23)$$

Тем самым для y_3 и y_2 остаются чисто алгебраические уравнения (21) и (23), и вместо системы ОДУ остается одно единственное дифференциальное уравнение для y_1

$$\dot{y}_1 = -k_1 y_1 + k_2(1 - y_1) \cdot \left[\frac{k_2(y_1 - 1)}{2k_3} + \frac{1}{2}\sqrt{\frac{4y_1 \cdot k_1}{k_3} + \left(\frac{k_2}{k_3(1 - y_1)}\right)^2}\right] \quad (24)$$

Оно легко решается численными методами. В этом случае простейший одношаговый метод Эйлера без какого-либо анализа устойчивости и регулирования шага дает наибольшую вычислительную эффективность. Никакой другой алгоритм не сможет превзойти результат 0,37 сек. вычислительного времени, затраченного на домашнем настольном компьютере. Суммарные итоги решения «задачи в два действия» представлены на рисунках 8–9.

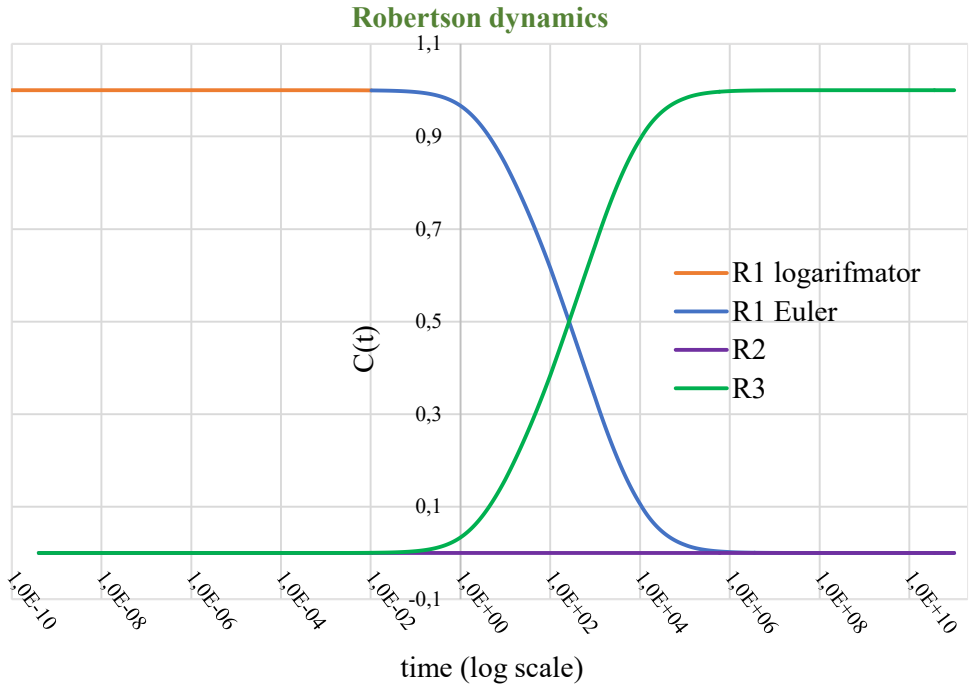

Рис. 8 Полное решение системы Робертсона на всем диапазоне времен от $1 \cdot 10^{-10}$ сек до $1 \cdot 10^{+11}$сек. Затрачено машинного времени 0,37 сек.

Fig.8. The complete solution of the Robertson system over the entire time range from 1 · 10 ^ (-10) seconds to 1 · 10 ^ (+11) seconds. Machine Time 0.37 sec

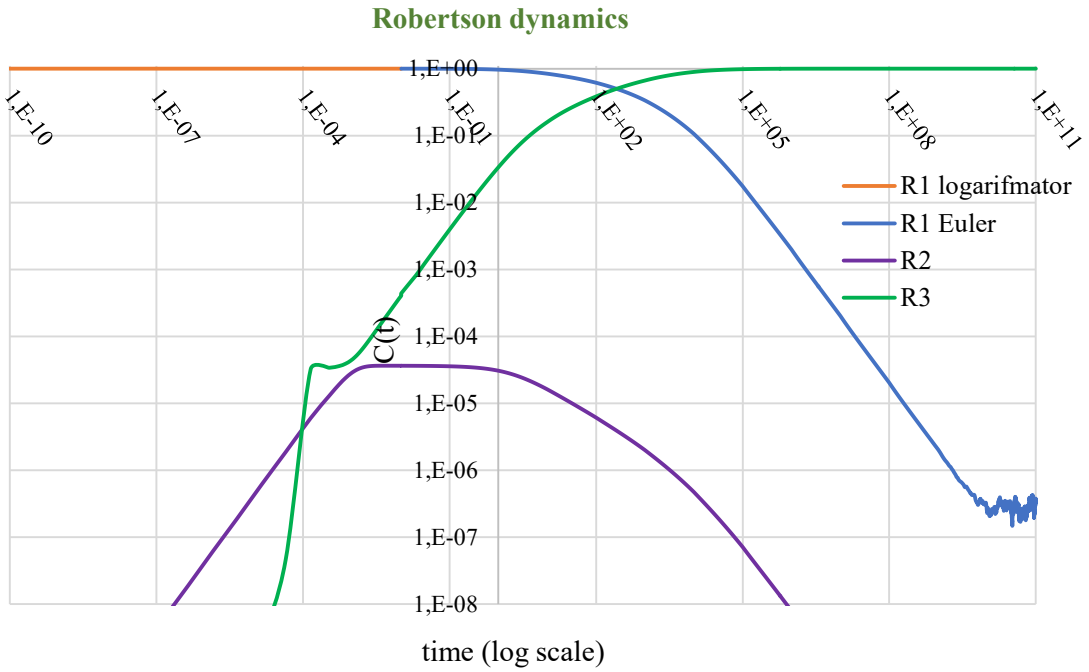

Рис. 9. То же, что и на рис.8, но логарифмический масштаб применен не только к оси абсцисс(время), но и к оси ординат (концентрации). Затрачено машинного времени 0,37 сек.

Fig.9. The same as in Fig. 8, but the logarithmic scale is applied not only to the abscissa axis (time), but also to the ordinate axis (concentration). Machine time spent 0.37 sec.

Видно, что на больших временах (точнее на малых остаточных концентрациях) алгоритм вновь начинает проявлять склонность к осцилляциям (зашумлению).

ЗАКЛЮЧЕНИЕ

Проведенный сравнительный анализ различных алгоритмов решения СОДУ показал неоспоримое преимущество неявных методов над явными, а многошаговых над одношаговыми. При этом само по себе использование «устойчивых» алгоритмов не гарантирует устойчивости реализуемого метода в случае неправильного выбора шага.

Следует также отметить, что вычислительные затраты, связанные с регулированием и оптимизацией шага по ходу интегрирования могут привести к потерям времени даже более значительным, по сравнению с изначальным выбором нерегулируемого, но достаточно мелкого шага.

Наилучшие же результаты дает предварительный экспертный анализ СОДЕ, после чего требование к устойчивости алгоритмов становится существенно более мягким.

КОНФЛИКТ ИНТЕРЕСОВ

Авторы заявляют об отсутствии конфликта интересов.

CONFLICT OF INTERESTS

The authors declare no conflict of interests.

Список литературы:

1. Васильев Е. И., Васильева Т. А., Киселева М. Н. (2013). L-устойчивость мульти-неявных методов 8-го порядка для жестких систем дифференциальных уравнений. *Математическая физика и компьютерное моделирование*. *1*(18), 70–83.
2. Заусаев А.Ф. (2010). Разностные методы решения обыкновенных дифференциальных уравнений. Самара: Самарский гос. техн. ун-т.
3. Chen, R. T., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. (2018). Neural ordinary differential equations. *Advances in neural information processing systems*, 31.
4. Hairer, E., Wanner, G., & Nørsett, S. P. (1993). Solving ordinary differential equations I: Nonstiff problems. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-78862-1.
5. Hairer, E., & Wanner, G. (1991). II: Stiff and differential-algebraic problems. Berlin [etc.]: Springer. https://doi.org/10.1007/978-3-662-09947-6.
6. Новиков Е.А. *Явные методы для жестких систем*. Новосибирск: Наука СО РАН, 1997, 195 с.
7. Robertson, H. H. (1966). The solution of a set of reaction rate equations. *Numerical analysis: an introduction*, *178182*, 31.
8. Новиков Е. А., Захаров А. А. (2015). *Алгоритм переменного шага с применением метода типа Розенброка третьего порядка точности*. Вестник Тюменского ГУ, Физико-математическое моделирование. Нефть, газ, энергетика, *1*(1), 146–154.
9. Гулевич Д. Р., Залипаев В. В. (2020). *Численные методы в физике и технике*. СПб.: УИТМО, 211 с.
10. Пинчуков В. И. (2002). Сравнение неявных схем Рунге-Кутты третьего порядка. *Вычислительные технологии*, *7*(5), 44–57.
11. Kidger, P. (2022). On neural differential equations. *arXiv preprint arXiv:2202.02435*. (Ph.D. dissertation), Trinity: Mathematical Institute University of Oxford. https://doi.org/10.48550/arXiv.2202.02435
12. Lagaris, I. E., Likas, A., & Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, *9*(5), 987–1000. https://doi.org/10.1109/72.712178.
13. Zhang, T., Zhang, Y., & Ju, Y. (2020). A deep learning-based ODE solver for chemical kinetics. *arXiv preprint arXiv:2012.12654*.
14. Ji, W., Qiu, W., Shi, Z., Pan, S., & Deng, S. (2021). Stiff-pinn: Physics-informed neural network for stiff chemical kinetics. *The Journal of Physical Chemistry A*, *125*(36), 8098–8106. https://doi.org/10.1021/acs.jpca.1c05102.
15. Gomes, C., Thule, C., Broman, D., Larsen, P. G., & Vangheluwe, H. (2018). Co-simulation: a survey. ACM Computing Surveys (CSUR), *51*(3), 1–33. https://doi.org/10.1145/3179993.

References:

1. Vasilyev, E. I., Vasilyeva, T. A., & Kiseleva M.N. (2013). L-stability of multi-implicit methods of 8-th order for differential stiff systems. *Mathematical Physics and Computer Modeling*. *1*(18), 70–83. (in Russ.).
2. Zausaev A. F. (2010). Difference methods for solving ordinary differential equations. Samara: Samara State Technical University. (in Russ.).
3. Chen, R. T., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. (2018). Neural ordinary differential equations. *Advances in neural information processing systems*, 31.
4. Hairer, E., Wanner, G., & Nørsett, S. P. (1993). Solving ordinary differential equations I: Nonstiff problems. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-78862-1.
5. Hairer, E., & Wanner, G. (1991). II: Stiff and differential-algebraic problems. Berlin [etc.]: Springer. https://doi.org/10.1007/978-3-662-09947-6.

6. Novikov E. A. Explicit Methods for Stiff Systems. Novosibirsk: Science SB RAS, 1997, 195 p. (in Russ.).
7. Robertson, H. H. (1966). The solution of a set of reaction rate equations. *Numerical analysis: an introduction*, *178182*, 31. (in Russ.).
8. Novikov E. A., & Zakharov A .A. (2015). Variable structure algorithm with the rosenbrock method of a third-order approximation applied. *Bulletin of Tyumen State University, Physical and Mathematical Modeling. Oil, Gas, Energy, 1*(1), 146–154. (in Russ.)
9. Gulevich, D. R., & Zalipaev, V.V. (2020). Numerical Methods in Physics and Engineering. St. Petersburg: UITMO, 211 p. (in Russ.).
10. Pinchukov, V. I. (2002). On a numerical solution of equations of a viscous gas by a third-order implicit Runge–Kutta scheme. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, *42*(6), 896–904.
11. Kidger, P. (2022). On neural differential equations. *arXiv preprint arXiv:2202.02435*. (Doctoral thesis), Trinity: Mathematical Institute University of Oxford. https://doi.org/10.48550/arXiv.2202.02435.
12. Lagaris, I. E., Likas, A., & Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, *9*(5), 987–1000. https://doi.org/10.1109/72.712178.
13. Zhang, T., Zhang, Y., & Ju, Y. (2020). A deep learning-based ODE solver for chemical kinetics. *arXiv preprint arXiv:2012.12654*.
14. Ji, W., Qiu, W., Shi, Z., Pan, S., & Deng, S. (2021). Stiff-pinn: Physics-informed neural network for stiff chemical kinetics. *The Journal of Physical Chemistry A*, *125*(36), 8098–8106. https://doi.org/10.1021/acs.jpca.1c05102.
15. Gomes, C., Thule, C., Broman, D., Larsen, P. G., & Vangheluwe, H. (2018). Co-simulation: a survey. ACM Computing Surveys (CSUR), *51*(3), 1–33. https://doi.org/10.1145/3179993.

Listing of VBA-Excel program

Public Sub Start_Heun()	
1.	Dim Y_ As Variant, P_ As Variant, k_ As Variant, bufer As Variant
2.	Dim Rg As Range, nrows As Integer, ncols As Integer
3.	Dim Rg_R As Range, Rg_L As Range, Rg_0 As Range, Rg_t As Range, RG_i As Range, RG_h As Range
4.	Dim threshold_ As Double, krator_ As Double, alfa_ As Double, maxstep_ As Double
5.	Dim i As Long, j As Integer, niter_ As Long, dTime As Double
6.	Application.Calculation = xlCalculationManual
7.	dTime = MicroTimer
8.	Set Rg = Range("Constants")
9.	nrows = Rg.Rows.Count
10.	ncols = Rg.Columns.Count
11.	Set Rg_0 = Range("Initio")
12.	niter_ = [niter]
13.	ReDim bufer(1, 1 To ncols)
14.	ReDim Y_(0 To niter_, 1 To ncols)
15.	ReDim P_(0 To niter_, 1 To ncols)
16.	ReDim t_points(0 To niter_, 1 To 1)

```
17.  ReDim hhh(0 To niter_, 1 To 1)
18.  ReDim k_(1 To 1, 1 To ncols)
19.  ReDim steps_(0 To 200, 1)
20.  steps_ = Range("Steps").Value
21.  k_ = Rg.Value
22.  bufer = Rg_0.Value
23.  threshold_ = Range("threshold").Value
24.  krator_ = Range("krator").Value
25.  alfa_ = Range("alfa")
26.  maxstep_ = Range("max_step")
     
27.  Set Rg_R = Range(Cells(Rg_0.Row, Rg_0.Column), Cells(Rg_0.Row + niter_, Rg_0.Column +
     ncols - 1))
28.  Set Rg_t = Range(Cells(Rg_0.Row, 1), Cells(Rg_0.Row + niter_, 1))
29.  Set RG_i = Range(Cells(Rg_0.Row, Rg_0.Column +2* ncols), Cells(Rg_0.Row + niter_,
     Rg_0.Column + 3*ncols - 1))
30.  Set RG_h = Range(Cells(Rg_0.Row, Rg_0.Column + 3 * ncols), Cells(Rg_0.Row + niter_,
     Rg_0.Column + 3 * ncols))
     
31.  t_points(0, 1) = 0
32.  For j = 1 To ncols
33.      Y_(0, j) = bufer(1, j)
34.  Next j
35.  For i = 0 To niter_
36.      On Error GoTo otskok
37.      If i < 20 Then
38.          hh = steps_(i + 1, 1)
39.      End If
40.      hhh(i, 1) = hh
41.      t_points(i + 1, 1) = t_points(i, 1) + hh
42.      bufer = PrCh(Y_, i, k_, ncols)
43.      For j = 1 To ncols
44.          P_(i, j) = bufer(1, j) * hh 'increments(i,j)
45.          Y_(i + 1, j) = Y_(i, j) + alfa_ * P_(i, j)
46.      Next j
47.      bufer = PrCh(Y_, i + 1, k_, ncols)
     
48.      For j = 1 To ncols
49.      Y_(i + 1, j) = Y_(i, j) + (1 - 1 / 2 / alfa_) * P_(i, j) + 1 / 2 / alfa_ * hh * bufer(1, j)
50.      Next j
51.      ' Regulation of step; after first 20
52.      If i > 20 Then
53.          If i Mod 5 = 0 Then
54.              If Abs(1 - (P_(i, 2) / P_(i - 1, 2))) > threshold_ Then
```

55.	hh = hh / krator_
56.	Else
57.	If Abs(1 - (P_(i, 2) / P_(i - 1, 2))) < threshold_ / krator_ Then
58.	hh = WorksheetFunction.Min(hh * krator_, maxstep_)
59.	End If
60.	End If
61.	End If
62.	End If
63.	Next i
	Rg_R.Value = Y_
	Rg_t.Value = t_points
	RG_i.Value = P_
	RG_h.Value = hhh
	[Q16] = Round((MicroTimer - dTime), 2)
	Application.Calculation = xlCalculationAutomatic