Полное руководство по численным методам решения ОДУ

Теория, алгоритмы и практическая реализация с подробными объяснениями

Профессор И.В. Вычислительный
Кафедра вычислительной математики и математической физики

Версия 2.1
5 декабря 2025 г.

Оглавление

Предисловие

Данное руководство представляет собой полное изложение теории численных методов решения обыкновенных дифференциальных уравнений (ОДУ) с подробными объяснениями на каждом этапе. Цель пособия — не только дать формальные определения и формулы, но и помочь читателю сформировать глубокую интуицию о том, **что происходит на каждом этапе, почему методы работают именно так, и как выбирать подходящий метод для конкретной задачи**.

Каждая глава содержит:

- Формальные математические определения
- Подробные текстовые объяснения на русском языке
- Геометрические интерпретации
- Практические рекомендации
- Примеры кода на Python

Руководство предназначено для студентов, аспирантов, исследователей и инженеров, которые хотят не только использовать численные методы, но и понимать их внутреннюю логику.

Глава 1

Математические основы задачи Коши

1.1 Постановка задачи

1.1.1 Формальная постановка

Рассмотрим задачу Коши для системы ОДУ первого порядка:

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0, \tag{1.1}$$

где:

- $\mathbf{y}(t) \in \mathbb{R}^n$ — вектор-функция искомого решения,
- $\mathbf{f} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ — заданная вектор-функция,
- $t \in [t_0, T]$ — независимая переменная,
- $\mathbf{y}_0 \in \mathbb{R}^n$ — начальное условие.

1.1.2 Что происходит на самом деле? Подробное объяснение

Физическая интерпретация: Представьте, что $\mathbf{y}(t)$ описывает состояние некоторой системы во времени. Например:

- В механике: $\mathbf{y}$ может содержать координаты и скорости частиц
- В химии: $\mathbf{y}$ — концентрации реагентов
- В биологии: $\mathbf{y}$ — численности популяций
- В электротехнике: $\mathbf{y}$ — токи и напряжения в цепи

Функция $\mathbf{f}(t, \mathbf{y})$ задает **правила изменения** системы. Она отвечает на вопрос: "Если в момент времени t система находится в состоянии $\mathbf{y}$, то с какой скоростью и в каком направлении она будет меняться?"

Начальное условие $\mathbf{y}_0$ — это "фотография"системы в начальный момент. Без этого условия задача не имеет единственного решения, так как из одной точки можно выходить по разным интегральным кривым.

Пример из физики: Второй закон Ньютона $m\frac{d^2\mathbf{x}}{dt^2} = \mathbf{F}(t, \mathbf{x}, \frac{d\mathbf{x}}{dt})$ можно записать в виде (1.1), если ввести $\mathbf{y} = (\mathbf{x}, \mathbf{v})$, где $\mathbf{v} = \frac{d\mathbf{x}}{dt}$.

1.2 Теорема существования и единственности

Теорема 1 (Пикара-Линделёфа)**.** *Пусть функция* $\mathbf{f}(t, \mathbf{y})$ *определена и непрерывна в области*

$$D = \{(t, \mathbf{y}) : |t - t_0| \leq a, \|\mathbf{y} - \mathbf{y}_0\| \leq b\},$$

и удовлетворяет условию Липшица по $\mathbf{y}$*:*

$$\|\mathbf{f}(t, \mathbf{y}_1) - \mathbf{f}(t, \mathbf{y}_2)\| \leq L\|\mathbf{y}_1 - \mathbf{y}_2\|$$

для всех $(t, \mathbf{y}_1), (t, \mathbf{y}_2) \in D$*. Тогда существует единственное решение задачи (1.1) на отрезке* $[t_0, t_0 + \alpha]$*, где*

$$\alpha = \min\left(a, \frac{b}{M}\right), \quad M = \max_{(t,\mathbf{y})\in D} \|\mathbf{f}(t, \mathbf{y})\|.$$

1.2.1 Интуитивное понимание теоремы

Зачем нужна эта теорема? Она гарантирует, что наша задача имеет смысл и мы можем пытаться ее решать численно.

Условие непрерывности: Функция $\mathbf{f}$ не должна иметь разрывов. Представьте, что вы ведете машину по дороге — если правила дорожного движения (функция $\mathbf{f}$) меняются скачком, вы не сможете плавно ехать.

Условие Липшица: Это самое важное условие. Оно означает, что функция $\mathbf{f}$ не может меняться **слишком резко**. Константа Липшица L ограничивает "скорость реакции"системы на изменение состояния.

Геометрический смысл условия Липшица: Если нарисовать график $\mathbf{f}$ как функции от $\mathbf{y}$, то угловой коэффициент любой секущей не может превышать L. Это предотвращает ситуации, когда решение "убегает на бесконечность за конечное время".

Почему важно единственность? Если бы решение не было единственным, то маленькие погрешности в начальных данных или вычислениях могли бы приводить к совершенно разным результатам, и численные методы были бы бесполезны.

Величина α**:** Это максимальная длина отрезка, на котором мы гарантируем существование решения. Она ограничена двумя факторами:

1. a — как далеко мы можем уйти по времени от t_0
2. b/M — как далеко мы можем уйти от начального состояния, учитывая максимальную скорость изменения

1.3 Линеаризация и матрица Якоби

Для анализа локального поведения системы вводим матрицу Якоби:

$$J(t, \mathbf{y}) = \frac{\partial \mathbf{f}}{\partial \mathbf{y}} = \begin{pmatrix} \frac{\partial f_1}{\partial y_1} & \cdots & \frac{\partial f_1}{\partial y_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial y_1} & \cdots & \frac{\partial f_n}{\partial y_n} \end{pmatrix} \tag{1.2}$$

Линеаризованная система в окрестности решения $\mathbf{y}^*(t)$:

$$\frac{d\mathbf{z}}{dt} = J(t, \mathbf{y}^*(t))\mathbf{z}, \quad \mathbf{z} = \mathbf{y} - \mathbf{y}^*(t) \tag{1.3}$$

1.3.1 Что такое матрица Якоби и зачем она нужна?

Матрица Якоби — это "многомерная производная": Если для функции одной переменной производная показывает, как быстро меняется функция при изменении аргумента, то для векторной функции $\mathbf{f}$ матрица Якоби показывает, как каждая компонента f_i реагирует на изменение каждой компоненты y_j.

Структура матрицы: Элемент $J_{ij} = \frac{\partial f_i}{\partial y_j}$ отвечает на вопрос: "Если немного изменить y_j, то как изменится скорость изменения y_i?"

Физическая аналогия: Представьте химическую реакцию с несколькими реагентами. Матрица Якоби показывает, как скорость образования каждого продукта зависит от концентрации каждого реагента.

Практическое применение:

1. **Анализ устойчивости:** Собственные значения матрицы Якоби определяют, будет ли решение устойчивым
2. **Численные методы:** В неявных методах (Ньютона) матрица Якоби используется для решения нелинейных уравнений
3. **Понимание системы:** Матрица Якоби помогает понять, какие переменные сильно влияют друг на друга

Линеаризация: В окрестности известного решения $\mathbf{y}^*(t)$ сложную нелинейную систему можно приблизить линейной. Это как аппроксимировать кривую касательной в точке — локально это хорошее приближение.

Глава 2

Основные понятия численных методов

2.1 Дискретизация и сетка

2.1.1 Формальное определение

Введем равномерную сетку:

$$t_n = t_0 + nh, \quad n = 0, 1, \ldots, N, \quad h = \frac{T - t_0}{N} \tag{2.1}$$

Обозначим:

- y_n — приближенное решение в точке t_n,
- $y(t_n)$ — точное решение в точке t_n,
- $e_n = y_n - y(t_n)$ — глобальная погрешность.

2.1.2 Процесс дискретизации — что мы на самом деле делаем?

От непрерывного к дискретному: Исходная задача непрерывная — мы ищем функцию $y(t)$, определенную для **всех** t на отрезке $[t_0, T]$. Численные методы превращают эту задачу в конечномерную — мы ищем **конечный набор чисел** $y_0, y_1, \ldots, y_N$.

Аналогия: Представьте, что вы снимаете видео (непрерывный процесс) и сохраняете его как последовательность кадров (дискретное представление). Чем больше кадров в секунду (меньше шаг h), тем точнее видео передает реальность, но тем больше места оно занимает (больше вычислений).

Шаг интегрирования h — главный параметр:

- **Слишком маленький h:** Высокая точность, но огромные вычислительные затраты. Это как измерять температуру каждую секунду — точность максимальная, но непрактично.
- **Слишком большой h:** Быстрое вычисление, но возможна большая погрешность или даже неустойчивость. Это как измерять температуру раз в день — можно пропустить важные изменения.

- **Оптимальный h:** Баланс между точностью и скоростью. Часто выбирается адаптивно — делаем мелкие шаги там, где решение меняется быстро, и крупные там, где оно меняется медленно.

Три вида погрешности:

1. **Локальная погрешность:** Ошибка, сделанная на одном шаге
2. **Глобальная погрешность:** Накопленная ошибка к моменту t_N
3. **Ошибка округления:** Из-за конечной точности компьютера

Важное наблюдение: Даже если на каждом шаге мы делаем маленькую ошибку (локальная погрешность), после многих шагов глобальная погрешность может стать большой. Это как идти по компасу: небольшое отклонение на каждом шаге через километр может привести в совершенно другую точку.

2.2 Классификация методов

1. **Одношаговые методы:** $y_{n+1} = \Phi(t_n, y_n, h)$
2. **Многошаговые методы:** $y_{n+1} = \Phi(t_n, \ldots, t_{n-k}, y_n, \ldots, y_{n-k}, h)$
3. **Явные методы:** y_{n+1} выражается явно через известные значения
4. **Неявные методы:** y_{n+1} входит в правую часть уравнения

2.2.1 Выбор метода — какая разница и что выбрать?

Одношаговые vs многошаговые:

Одношаговые (Эйлер, Рунге-Кутта)	**Многошаговые (Адамса, BDF)**
• Используют только y_n • Легко стартовать (y_0 известно) • Можно легко менять шаг h • Пример: метод Рунге-Кутта	• Используют $y_n, y_{n-1}, \ldots$ • Требуют "разгона"(первые точки другим методом) • При фиксированном h более эффективны • Пример: метод Адамса

Явные vs неявные:

Явные методы	Неявные методы
• y_{n+1} вычисляется по явной формуле • Простые в реализации • Могут быть неустойчивыми • Пример: явный Эйлер $y_{n+1} = y_n + hf(t_n, y_n)$	• y_{n+1} входит в уравнение • Требуют решения нелинейного уравнения • Более устойчивы, особенно для жестких систем • Пример: неявный Эйлер $y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$

Практические рекомендации по выбору:

- **Для нежестких задач:** Явные методы Рунге-Кутты высокого порядка
- **Для жестких задач:** Неявные методы (BDF, неявный Рунге-Кутта)
- **Если вычисление f дорогое:** Многошаговые методы (меньше вычислений f на шаг)
- **Если нужно часто менять шаг:** Одношаговые методы

2.3 Понятие порядка точности

Определение 1 (Локальная погрешность)**.** *Локальной погрешностью метода называется величина:*

$$\delta_{n+1} = y(t_{n+1}) - \tilde{y}_{n+1},$$

где $\tilde{y}_{n+1}$ — значение, полученное по методу из точного начального условия $y(t_n)$.

Определение 2 (Порядок точности)**.** *Метод имеет порядок точности p, если его локальная погрешность удовлетворяет:*

$$\delta_{n+1} = O(h^{p+1}) \quad при \quad h \to 0.$$

2.3.1 Что означает порядок точности на практике?

Простая аналогия: Представьте, что вы приближаете кривую отрезками.

- **Порядок 1 (Эйлер):** Аппроксимируем кривую ломаной. Уменьшение шага в 2 раза уменьшает ошибку в 2 раза.
- **Порядок 4 (RK4):** Аппроксимируем кривую кубическими сплайнами. Уменьшение шага в 2 раза уменьшает ошибку в 16 раз.

Формально: Если метод имеет порядок p, то:

$$\text{Глобальная погрешность} \approx C \cdot h^p$$

где C — некоторая константа, не зависящая от h.

Как это работает:

1. **Локальная погрешность:** На одном шаге мы делаем ошибку порядка h^{p+1}
2. **Количество шагов:** На отрезке длины $T - t_0$ делаем $N \sim 1/h$ шагов
3. **Глобальная погрешность:** Примерно $N \cdot h^{p+1} \sim h^p$

Наглядный пример:

Метод	**Порядок**	**Что происходит при** $h \to h/2$
Явный Эйлер	1	Погрешность $\to$ погрешность/2
RK2	2	Погрешность $\to$ погрешность/4
RK4	4	Погрешность $\to$ погрешность/16

Важное замечание: Высокий порядок — не всегда хорошо:

- **Плюсы:** Можно брать большие шаги, меньше вычислений
- **Минусы:** Методы высокого порядка часто менее устойчивы, требуют больше памяти, сложнее в реализации

Правило выбора: Используйте методы порядка 4-5 для большинства нежестких задач. Это оптимальный компромисс между точностью, устойчивостью и сложностью.

2.4 Устойчивость численных методов

2.4.1 Тестовое уравнение

Для анализа устойчивости используется модельное уравнение:

$$y' = \lambda y, \quad \lambda \in \mathbb{C}, \quad \mathrm{Re}(\lambda) < 0 \tag{2.2}$$

Точное решение: $y(t) = y_0 e^{\lambda t} \to 0$ при $t \to \infty$.

2.4.2 Область устойчивости

Определение 3. *Областью абсолютной устойчивости метода называется множество комплексных чисел $z = h\lambda$, для которых численное решение тестового уравнения стремится к нулю при $n \to \infty$.*

2.4.3 Устойчивость — почему это важно и как это работает?

Проблема: Можно придумать метод с очень высоким порядком точности, но который дает бессмысленные результаты из-за неустойчивости. Это как построить очень красивый мост, который развалится от легкого ветра.

Тестовое уравнение $y' = \lambda y$ **— "лабораторная крыса":**

- **Почему именно это уравнение?** Оно простое, но захватывает ключевую особенность — экспоненциальное затухание или рост

- **Точное решение:** $y(t) = y_0 e^{\lambda t}$
- **При** $\mathrm{Re}(\lambda) < 0$**:** Решение затухает экспоненциально
- **При** $\mathrm{Re}(\lambda) > 0$**:** Решение растет экспоненциально

Что мы проверяем: Будет ли численный метод правильно воспроизводить затухание, когда точное решение затухает?

Область устойчивости — геометрическая интерпретация:

- Берем все возможные $z = h\lambda$ такие, что $\mathrm{Re}(\lambda) < 0$
- Для каждого z смотрим: стремится ли численное решение к 0?
- Область устойчивости — множество тех z, для которых ответ "да"

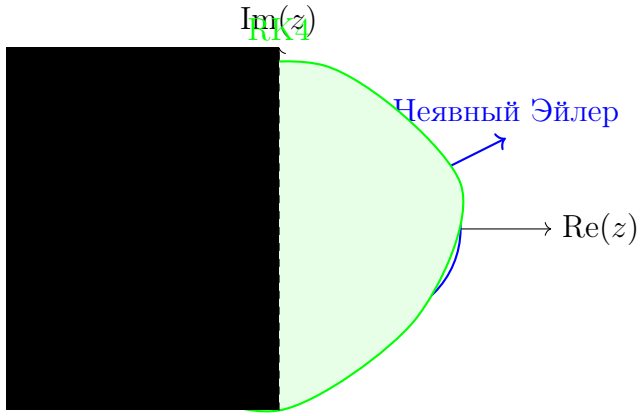


Рис. 2.1: Области устойчивости различных методов (схематично). Для A-устойчивости область должна содержать всю левую полуплоскость.

Типы устойчивости:

1. **A-устойчивость:** Область устойчивости содержит всю левую полуплоскость $\mathrm{Re}(z) < 0$
2. **L-устойчивость:** A-устойчивость + $|R(z)| \to 0$ при $\mathrm{Re}(z) \to -\infty$
3. **Условная устойчивость:** Область устойчивости ограничена

Практические следствия:

- **Для жестких систем:** Нужны A-устойчивые методы
- **Для нежестких систем:** Достаточно методов с большой областью устойчивости
- **Выбор шага:** Для условно устойчивых методов шаг h должен быть таким, чтобы $h\lambda$ попадал в область устойчивости

Важное наблюдение: Неявные методы обычно имеют бóльшие области устойчивости, чем явные. Это одна из причин, почему они используются для жестких систем.

Глава 3

Одношаговые методы

3.1 Метод Эйлера: полный вывод

3.1.1 Явный метод Эйлера

Разложим точное решение в ряд Тейлора в точке t_n:

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(\xi_n), \quad \xi_n \in [t_n, t_{n+1}] \tag{3.1}$$

Используя $y'(t_n) = f(t_n, y(t_n))$, получаем:

$$y(t_{n+1}) = y(t_n) + hf(t_n, y(t_n)) + \frac{h^2}{2}y''(\xi_n) \tag{3.2}$$

Отбрасывая остаточный член, приходим к явному методу Эйлера:

$$y_{n+1} = y_n + hf(t_n, y_n) \tag{3.3}$$

Локальная погрешность: $\delta_{n+1} = \frac{h^2}{2}y''(\xi_n) = O(h^2)$. Глобальная погрешность: $e_n = O(h)$ — метод первого порядка.

3.1.2 Геометрическая интерпретация метода Эйлера

Представьте, что вы идете по горной тропе в тумане:

- Вы стоите в точке A с координатами (t_n, y_n)
- Вы видите только направление прямо перед собой — это $f(t_n, y_n)$ (наклон траектории)
- Вы делаете шаг длины h в этом направлении
- Попадаете в точку B с координатами (t_{n+1}, y_{n+1})

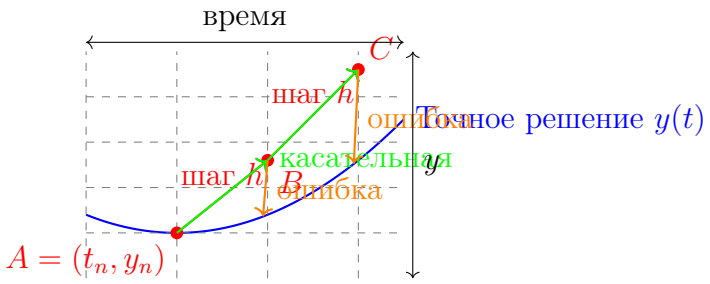


Рис. 3.1: Геометрическая интерпретация метода Эйлера. Красная ломаная — численное решение, синяя кривая — точное решение.

Что происходит на каждом шаге:

1. **В точке A:** Вычисляем направление $f(t_n, y_n)$
2. **Двигаемся:** Идем h единиц времени в этом направлении
3. **Ошибка:** Поскольку направление меняется вдоль кривой, мы отклоняемся от точного решения

Почему метод первого порядка?

- **Локально:** Мы аппроксимируем кривую отрезком касательной
- **Ошибка на шаге:** Пропорциональна h^2 (кривизна $\times$ h^2)
- **Глобальная ошибка:** Накопление $N \sim 1/h$ ошибок дает $O(h)$

Когда метод работает хорошо:

- Когда решение почти линейное (малая кривизна)
- Когда шаг h очень мал
- Для грубых оценок и учебных задач

Когда метод плох:

- Для решений с большой кривизной
- Для жестких систем (требует очень малых шагов)
- Когда нужна высокая точность

3.1.3 Неявный метод Эйлера

Разложим в ряд Тейлора в точке t_{n+1}:

$$y(t_n) = y(t_{n+1}) - hy'(t_{n+1}) + \frac{h^2}{2}y''(\xi_n) \tag{3.4}$$

Отсюда:

$$y(t_{n+1}) = y(t_n) + hf(t_{n+1}, y(t_{n+1})) - \frac{h^2}{2}y''(\xi_n) \tag{3.5}$$

Получаем неявный метод:

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}) \tag{3.6}$$

3.1.4 Неявный Эйлер — в чем философия?

Аналогия с навигацией:

- **Явный Эйлер:** "Куда идти?"Смотрю компас и иду в том направлении.
- **Неявный Эйлер:** "Где я должен оказаться?"Выбираю точку так, чтобы если я из нее смотрю назад, то вижу направление на свою текущую позицию.

Математически: Вместо того чтобы использовать направление в начале шага, мы используем направление в конце шага. Но проблема в том, что мы не знаем, где конец шага!

Как решать уравнение $y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$?

1. Записываем $F(y) = y - y_n - hf(t_{n+1}, y) = 0$
2. Применяем метод Ньютона: $y^{(k+1)} = y^{(k)} - J^{-1}F(y^{(k)})$
3. Итерируем до сходимости

Преимущества неявного метода:

- **Более устойчив:** Особенно для жестких систем
- **Может брать большие шаги:** Не так ограничен условиями устойчивости
- **Лучше для "крутых"решений:** Учитывает изменение направления на шаге

Недостатки:

- **Сложнее реализовать:** Нужно решать нелинейные уравнения
- **Дороже вычислять:** Каждый шаг требует нескольких вычислений f и решения системы
- **Может не сходиться:** Если начальное приближение плохое или шаг слишком большой

Геометрическая интерпретация:

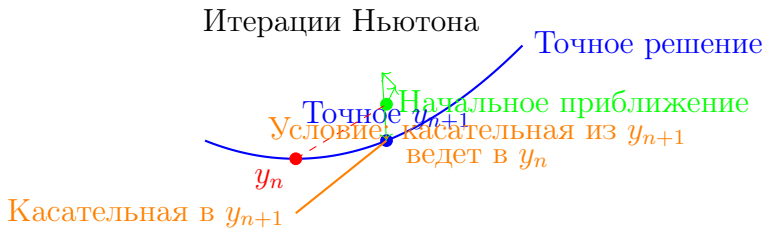


Рис. 3.2: Неявный метод Эйлера: итерационное уточнение y_{n+1}

3.2 Семейство методов Рунге-Кутты

3.2.1 Философия методов Рунге-Кутты

Основная идея: Вместо того чтобы использовать одно значение производной (как Эйлер), вычисляем производную в нескольких точках на шаге и берем взвешенное среднее.

Аналогия с измерением уклона дороги:

- **Эйлер:** Измеряю уклон в начале участка, иду все время с этим уклоном
- **RK2:** Измеряю уклон в начале, делаю полшага, измеряю уклон там, корректирую направление
- **RK4:** Измеряю уклон в начале, в середине (дважды), в конце, и вычисляю оптимальное среднее

Почему это работает лучше?

1. Учитываем изменение производной на шаге
2. Получаем более точную аппроксимацию интеграла
3. Можно достичь высокого порядка без использования предыдущих точек

Общая структура метода Рунге-Кутты:

$$
\begin{aligned}
k_1 &= hf(t_n, y_n) && \text{<– наклон в начале} \\
k_2 &= hf(t_n + c_2 h, y_n + a_{21} k_1) && \text{<– наклон в промежуточной точке} \\
k_3 &= hf(t_n + c_3 h, y_n + a_{31} k_1 + a_{32} k_2) && \text{<– еще одна точка} \\
&\vdots \\
y_{n+1} &= y_n + \sum b_i k_i && \text{<– взвешенное среднее}
\end{aligned}
$$

Ключевой вопрос: Как выбрать коэффициенты c_i, a_{ij}, b_i?

- Разлагаем и точное решение, и формулу метода в ряд Тейлора
- Приравниваем коэффициенты при одинаковых степенях h
- Получаем систему уравнений

3.2.2 Классический метод Рунге-Кутты 4-го порядка

$$
\begin{aligned}
k_1 &= hf(t_n, y_n) \\
k_2 &= hf(t_n + \tfrac{h}{2}, y_n + \tfrac{k_1}{2}) \\
k_3 &= hf(t_n + \tfrac{h}{2}, y_n + \tfrac{k_2}{2}) \\
k_4 &= hf(t_n + h, y_n + k_3) \\
y_{n+1} &= y_n + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

3.2.3 Геометрическая интерпретация RK4

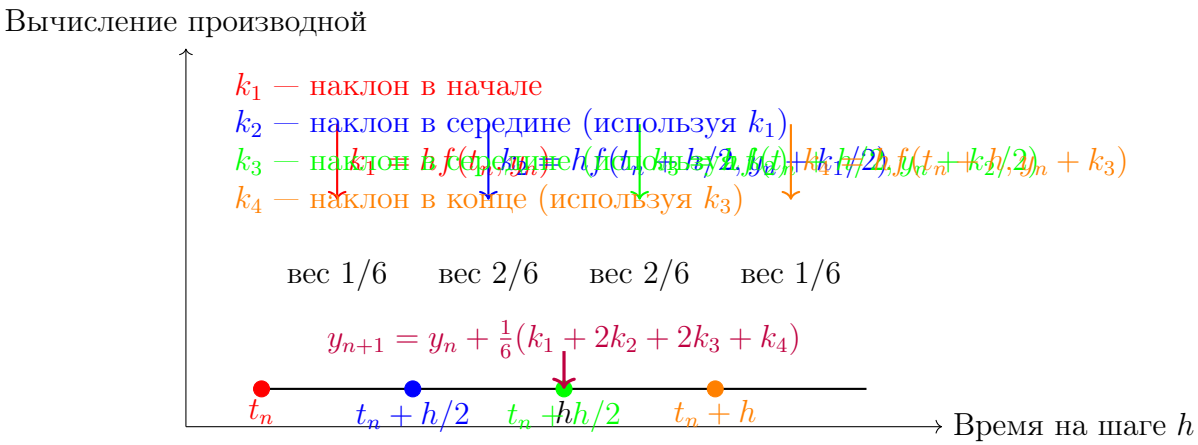


Рис. 3.3: Схема вычислений в методе Рунге-Кутты 4-го порядка

Что происходит на каждом шаге RK4:

1. **Пробный шаг 1 (k_1):** Смотрим, куда ведет касательная в начале
2. **Пробный шаг 2 (k_2):** Делаем полшага в направлении k_1, смотрим новое направление
3. **Пробный шаг 3 (k_3):** Делаем полшага в направлении k_2 (лучше, чем k_1), смотрим направление
4. **Пробный шаг 4 (k_4):** Делаем полный шаг в направлении k_3, смотрим направление в конце
5. **Итог:** Берем взвешенное среднее всех направлений

Почему веса именно такие (1/6, 2/6, 2/6, 1/6)? Эти веса получаются из условия, чтобы метод имел 4-й порядок точности. Они соответствуют весам в формуле Симпсона для численного интегрирования.

Аналогия с численным интегрированием: Решение ОДУ $y' = f(t, y)$ на шаге $[t_n, t_{n+1}]$:

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))dt$$

Метод Рунге-Кутты аппроксимирует этот интеграл, вычисляя f в нескольких точках.

Преимущества RK4:

- Высокий порядок точности (4) при разумных вычислительных затратах
- Самозапускающийся (не нужны предыдущие точки)
- Хорошая устойчивость для нежестких задач

Недостатки:

- Требует 4 вычисления f на шаг
- Не A-устойчив (плох для жестких систем)
- Нет встроенной оценки погрешности (нужны вложенные методы)

Глава 4

Многошаговые методы

4.1 Идея многошаговых методов

4.1.1 Философия многошаговых методов

Основная идея: Использовать уже вычисленную информацию о решении (значения y и f в предыдущих точках), чтобы строить более точные аппроксимации на следующем шаге.

Аналогия с предсказанием погоды:

- **Одношаговый метод:** Смотрю на погоду сейчас, предсказываю на завтра
- **Многошаговый метод:** Смотрю на погоду сегодня, вчера, позавчера... вижу тенденцию, предсказываю на завтра

Математическая основа: Интегральное представление решения:

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))dt$$

Вместо того чтобы аппроксимировать f на $[t_n, t_{n+1}]$ по одной точке (Эйлер) или нескольким точкам (Рунге-Кутта), мы строим интерполяционный полином по значениям f в нескольких предыдущих точках и интегрируем его аналитически.

Пример интерполяции:

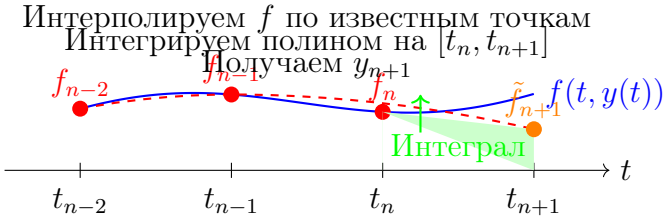


Рис. 4.1: Идея методов Адамса: интерполяция f и интегрирование

Два основных класса многошаговых методов:

1. **Методы Адамса:** Интерполируют $f(t, y(t))$
2. **Методы BDF (Гира):** Интерполируют $y(t)$

Ключевые особенности:

- **Эффективность:** На шаг требуется только 1 вычисление f (после разгона)
- **Высокий порядок:** Легко получать методы высокого порядка
- **Проблемы:** Нужен разгон, сложно менять шаг

4.2 Методы Адамса

4.2.1 Явные методы Адамса-Башфорта

Интерполяция по k предыдущим точкам:

$$y_{n+1} = y_n + h \sum_{j=0}^{k-1} \beta_j f_{n-j} \tag{4.1}$$

4.2.2 Как работают методы Адамса-Башфорта?

Процесс на шаге:

1. **Имеем:** Значения $y_n, y_{n-1}, \ldots$ и $f_n = f(t_n, y_n), f_{n-1}, \ldots$
2. **Строим:** Интерполяционный полином $P(t)$ степени $k-1$ по точкам (t_{n-j}, f_{n-j}), $j = 0, \ldots, k-1$
3. **Интегрируем:** $\int_{t_n}^{t_{n+1}} P(t)dt$
4. **Получаем:** Линейную комбинацию f_{n-j} с коэффициентами β_j

Формулы для малых k:

- $k = 1$: $y_{n+1} = y_n + hf_n$ (это явный Эйлер!)
- $k = 2$: $y_{n+1} = y_n + \frac{h}{2}(3f_n - f_{n-1})$
- $k = 3$: $y_{n+1} = y_n + \frac{h}{12}(23f_n - 16f_{n-1} + 5f_{n-2})$

Откуда берутся коэффициенты? Из условия, чтобы полином $P(t)$ точно интегрировал $f(t)$ для $f(t) = 1, t, t^2, \ldots$

Пример для $k = 2$:

1. Ищем полином $P(t) = a + b(t - t_n)$ по точкам (t_n, f_n) и (t_{n-1}, f_{n-1})
2. Получаем: $P(t) = f_n + \frac{f_n - f_{n-1}}{h}(t - t_n)$
3. Интегрируем: $\int_{t_n}^{t_{n+1}} P(t)dt = hf_n + \frac{h}{2}(f_n - f_{n-1}) = \frac{h}{2}(3f_n - f_{n-1})$

Проблема разгона: Для метода, использующего k предыдущих точек, нужны $y_0, y_1, \ldots, y_{k-1}$. Первые $k-1$ точек получают одношаговым методом (например, Рунге-Кутты).

4.3 Методы Гира (BDF)

4.3.1 Философия методов BDF

Идея, обратная методам Адамса: Вместо того чтобы интерполировать f и интегрировать, мы интерполируем саму функцию $y(t)$ и требуем, чтобы производная интерполяционного полинома в точке t_{n+1} равнялась $f(t_{n+1}, y_{n+1})$.

Математически:

1. Строим интерполяционный полином $Q(t)$ по точкам (t_{n+1}, y_{n+1}), (t_n, y_n), (t_{n-1}, y_{n-1}), ...
2. Требуем: $Q'(t_{n+1}) = f(t_{n+1}, y_{n+1})$
3. Получаем линейное соотношение между $y_{n+1}, y_n, y_{n-1}, \ldots$

Почему это хорошо для жестких систем?

- Методы BDF обычно более устойчивы, чем методы Адамса
- Особенно BDF1 и BDF2 имеют хорошие свойства устойчивости
- Хорошо работают, когда решение меняется медленно (можно брать большие шаги)

Пример BDF2:

- Строим квадратичный полином по точкам (t_{n+1}, y_{n+1}), (t_n, y_n), (t_{n-1}, y_{n-1})
- Вычисляем $Q'(t_{n+1})$
- Приравниваем к $f(t_{n+1}, y_{n+1})$
- Получаем: $\frac{3y_{n+1} - 4y_n + y_{n-1}}{2h} = f(t_{n+1}, y_{n+1})$
- Или: $y_{n+1} = \frac{4}{3}y_n - \frac{1}{3}y_{n-1} + \frac{2}{3}hf(t_{n+1}, y_{n+1})$

Неявность: Заметим, что y_{n+1} входит в обе части уравнения — метод неявный. Нужно решать нелинейное уравнение.

Области устойчивости BDF методов:

- BDF1 (неявный Эйлер): A-устойчив
- BDF2: A()-устойчив с 90°
- BDF3-BDF6: Устойчивость уменьшается с ростом порядка
- BDF6 и выше: Не рекомендуются из-за плохой устойчивости

Практическое использование: BDF методы — стандартный выбор для жестких систем в большинстве библиотек (MATLAB's ode15s, SciPy's BDF, SUNDIALS CVODE).

Глава 5

Жесткие системы ОДУ

5.1 Что такое жесткость и почему это проблема?

5.1.1 Интуитивное понимание жесткости

Аналогия с пружинами: Представьте систему из двух пружин:

- **Первая пружина:** Очень жесткая, быстро колеблется (частота 1000 Гц)
- **Вторая пружина:** Мягкая, медленно колеблется (частота 1 Гц)

Если вы хотите смоделировать движение такой системы, у вас проблема:

- Чтобы точно отслеживать первую пружину, нужен шаг 0.001 с
- Но вторая пружина почти не меняется за такое время
- Получается, вы делаете 1000 шагов, чтобы увидеть одно колебание второй пружины

Формальное определение: Система жесткая, если:

1. Есть компоненты решения, которые меняются очень быстро (затухают или осциллируют)
2. Есть компоненты, которые меняются медленно
3. Нас интересует поведение на временах, когда быстрые компоненты уже затухли

Число жесткости: $S = \frac{\max_i |\operatorname{Re}(\lambda_i)|}{\min_i |\operatorname{Re}(\lambda_i)|} \gg 1$

Пример из химии: Реакция с быстрым установлением равновесия и медленным основным процессом.

Почему явные методы плохи для жестких систем?

- Для устойчивости нужно $h < \frac{C}{\max |\lambda_i|}$ (очень маленький шаг)
- Но интересующее нас время моделирования определяется медленными компонентами
- Получаем огромное количество шагов для моделирования медленного процесса

Пример: Пусть $\lambda_1 = -1000$, $\lambda_2 = -1$. Для явного Эйлера нужно $h < 0.002$ для устойчивости. Чтобы проинтегрировать до $t = 10$, нужно 5000 шагов, хотя медленная компонента меняется на масштабе 1.

5.2 Методы для жестких систем

5.2.1 Почему неявные методы работают лучше?

Ключевое наблюдение: Неявные методы имеют бóльшие области устойчивости. В частности, неявный Эйлер A-устойчив — устойчив при любом h для задачи $y' = \lambda y$, $\mathrm{Re}(\lambda) < 0$.

Что это значит на практике: Для жесткой системы мы можем выбрать шаг h, исходя из точности, а не из устойчивости.

Пример с пружинами:

- **С явным методом:** $h \sim 0.001$ (определяется быстрой пружиной)
- **С неявным методом:** $h \sim 0.1$ (определяется точностью для медленной пружины)

Разница в 100 раз по количеству шагов!

Цена: Неявные методы требуют решения нелинейных уравнений на каждом шаге. Но часто это дешевле, чем делать тысячи шагов явным методом.

Сравнение затрат:

Явный метод (RK4)	**Неявный метод (BDF2)**
• 4 вычисления f на шаг • Простое векторное сложение • Нужно $N \sim 10000$ шагов • Итого: 40000 вычислений f	• 1 вычисление f на шаг • Решение системы (3 итерации Ньютона) • Нужно $N \sim 100$ шагов • Итерация: вычисление f + решение линейной системы • Итого: 300 вычислений f + решение систем

Когда неявные методы выгодны: Когда разница в количестве шагов компенсирует сложность шага.

Методы для жестких систем:

1. **Неявный Эйлер:** Простой, A-устойчивый, но низкого порядка
2. **Методы BDF:** Хороший компромисс между порядком и устойчивостью
3. **Неявные Рунге-Кутты:** Высокий порядок, но сложная реализация
4. **Методы Розенброка:** Полунеявные, проще чем полностью неявные

Практический совет: Если ваш решатель требует неоправданно маленьких шагов или выдает неустойчивое решение — попробуйте перейти на неявный метод.

Глава 6

Практические аспекты и реализация

6.1 Адаптивный выбор шага

6.1.1 Зачем нужен адаптивный шаг?

Проблема фиксированного шага:

- Если взять слишком большой шаг — большая погрешность или неустойчивость
- Если взять слишком маленький шаг — избыточные вычисления
- Решение может меняться с разной скоростью на разных участках

Идея адаптивного шага: Менять шаг h в процессе интегрирования:

- Увеличивать, когда решение меняется медленно
- Уменьшать, когда решение меняется быстро

Как оценить погрешность? Используем вложенные методы:

- Вычисляем два решения разного порядка: y_{n+1} (порядок p) и $\hat{y}_{n+1}$ (порядок $p+1$ или $p-1$)
- Разница $\Delta = \|y_{n+1} - \hat{y}_{n+1}\|$ оценивает погрешность
- Если Δ слишком велика — уменьшаем шаг и пересчитываем
- Если Δ слишком мала — увеличиваем шаг на следующем шаге

Пример: Метод Рунге-Кутты-Фельберга 4(5) (RK45):

- Основная формула: 5-го порядка
- Вложенная формула: 4-го порядка
- Используют одни и те же вычисления f
- Разница дает оценку погрешности

Стратегия выбора шага:

1. Задаем допустимую погрешность ε (абсолютную и относительную)
2. Вычисляем масштаб: $\text{scale} = \text{atol} + \text{rtol} \cdot |y|$

3. Вычисляем нормированную ошибку: $\text{err} = \|\Delta\|/\text{scale}$
4. Если $\text{err} \leq 1$ — шаг принят
5. Новый шаг: $h_{\text{new}} = h \cdot \min(2, \max(0.5, 0.9 \cdot \text{err}^{-1/(p+1)}))$

Почему коэффициент 0.9? Для "консервативности"— чтобы не делать слишком резких изменений шага.

Начальный шаг: Часто выбирают эвристически:

$$h_0 = 0.01 \cdot \frac{\|y_0\|}{\|f(t_0, y_0)\|}$$

или более сложные стратегии.

6.2 Решение нелинейных уравнений в неявных методах

6.2.1 Метод Ньютона для неявных методов

Проблема: Неявный метод дает уравнение $F(y_{n+1}) = 0$, где

$$F(y) = y - y_n - hf(t_{n+1}, y)$$

Метод Ньютона:

1. Начальное приближение: $y^{(0)}$ (часто из явного метода)
2. Итерация: $y^{(k+1)} = y^{(k)} - J^{-1}F(y^{(k)})$
3. Критерий остановки: $\|F(y^{(k)})\| < \varepsilon$

Матрица Якоби: $J = I - h\frac{\partial f}{\partial y}(t_{n+1}, y^{(k)})$

Проблемы:

- Вычисление Якоби дорого (особенно для больших систем)
- Решение линейной системы на каждой итерации
- Сходимость только при хорошем начальном приближении

Упрощения:

1. **Упрощенный Ньютон:** Используем одну и ту же матрицу Якоби несколько итераций или шагов
2. **Конечно-разностная аппроксимация:** Если аналитическая Якоби недоступна
3. **Итерации с фиксированной точкой:** Проще, но медленнее сходятся

Практические советы:

- Для маленьких систем можно вычислять Якоби аналитически

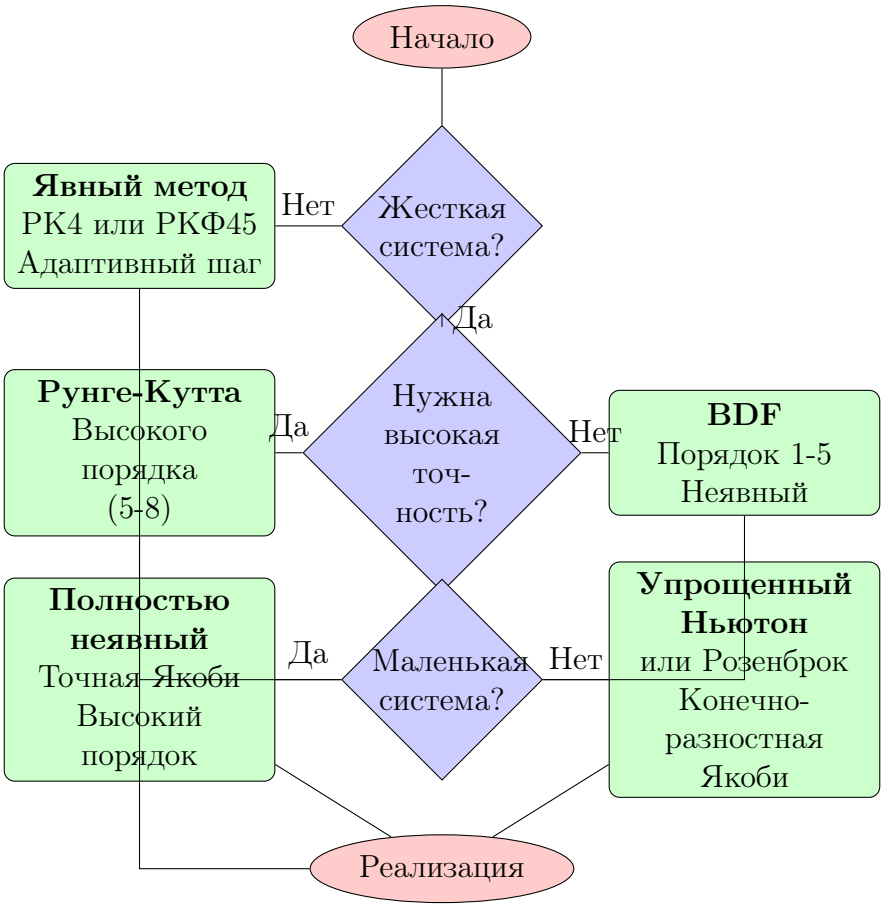
- Для больших систем — использовать конечно-разностные аппроксимации или автоматическое дифференцирование
- Использовать хорошие начальные приближения (предсказатель)
- Контролировать число итераций (ограничить максимум 3-5)

Критерии остановки:

- Абсолютный: $\|F(y^{(k)})\| < \varepsilon_a$
- Относительный: $\|F(y^{(k)})\| < \varepsilon_r \|y^{(k)}\|$
- Смешанный: $\|F(y^{(k)})\| < \varepsilon_a + \varepsilon_r \|y^{(k)}\|$

Заключение: как выбрать метод?

Практическое руководство по выбору метода:



Краткие рекомендации:

1. **Для большинства нежестких задач:** Используйте метод Рунге-Кутты 4-го или 5-го порядка с адаптивным выбором шага (например, Dormand-Prince 4(5)).
2. **Если вычисление f очень дорогое:** Рассмотрите многошаговые методы (Адамса) — они требуют только одно вычисление f на шаг после разгона.
3. **Для жестких систем:** Используйте неявные методы:
   - Простые задачи: Неявный Эйлер
   - Средние задачи: BDF методы (порядок 2-5)

- Сложные задачи: Неявные Рунге-Кутты или методы Розенброка

4. **Если не знаете, жесткая ли система:** Начните с явного метода. Если он требует очень маленьких шагов или неустойчив — переходите на неявный.
5. **Всегда контролируйте точность:** Используйте адаптивный выбор шага с оценкой погрешности.
6. **Проверяйте решение:** Сравнивайте с аналитическим решением (если есть), проверяйте сохранение инвариантов, тестируйте на разных шагах.
7. **Используйте готовые библиотеки:** Не пишите методы с нуля для серьезных задач. Используйте проверенные реализации:
   - Python: `scipy.integrate.solve_ivp`
   - MATLAB: `ode45`, `ode15s`
   - Julia: `DifferentialEquations.jl`
   - C++: `SUNDIALS`, `Boost.Odeint`

Заключительная мысль: Понимание теории численных методов решения ОДУ важно не только для реализации методов, но и для их грамотного использования. Знание сильных и слабых сторон каждого метода, понимание причин неустойчивости, умение оценивать погрешность — все это позволяет эффективно решать реальные задачи и интерпретировать полученные результаты.