Лекции по численным методам в механике космического полета

Широбоков М.Г.
ФПМИ МФТИ

12 апреля 2025 г.

Оглавление

Введение

Пособие содержит семестровый курс лекций по «Численным методам в механике космического полета», который автор много лет читает студентам 3-го курса физтех-школы прикладной математики и информатики МФТИ на кафедре математического моделирования и прикладной математики. Пособие посвящено как численным методам общего назначения (оптимизации, решению алгебраических и обыкновенных дифференциальных уравнений), так методам, специфическим для механики космического полета (регуляризации и стабилизации уравнений движения, выбору переменных модели). Курс представляет собой обзор наиболее часто используемых в механике космического полета численных методов и опирается на более чем десятилетний опыт работы автора в предметной области. Так, среди методов оптимизации основное внимание уделяется широко зарекомендовавшим в механике полета и теории управления методам последовательного квадратичного программирования и внутренней точки, существенно помогающим решать оптимизационные задачи, в частности, краевые задачи. Среди методов решения обыкновенных дифференциальных уравнений большой популярностью пользуются вложенные методы Рунге – Кутты на основе схем Дормана – Принса и особенно многошаговые методы Адамса с адаптивным шагом и переменного порядка. Описывается метод интегрирования с выходом на ограничение, выводятся общие формулы уравнений в вариациях. Приводятся методы решения недоопределенных и переопределенных систем, также очень частно встречающиеся в прикладных задачах. Рассматриваются методы регуляризации и стабилизации уравнений движения космических аппаратов, ускоряющих процесс численного интегрирования. В конце пособия приводятся задачи для самостоятельного решения. Замечания к тексту можно направлять на электронный адрес автора shirobokov@phystech.edu.

Программа курса

1. Производные типичных для механики космического полета скалярных и векторных функций. Методы решения систем линейных уравнений, недоопределенные и переопределенные системы. Псевдообратные матрицы Мура–Пенроуза.

2. Численное дифференцирование, правые и центральные разностные схемы и выбор оптимального шага дифференцирования. Метод дихотомии, метод простых итераций, метод Ньютона, метод градиентного спуска и условия их сходимости. Метод Левенберга – Марквардта и логика управления свободным параметром.

3. Необходимые и достаточные условия оптимальности в задачах математического программирования. Методы последовательного квадратичного программирования (SQP). Формула Бройдена – Флетчера – Гольдфарба – Шанно. Теорема о сходимости метода SQP. Методы внутренней точки. Методы активных ограничений и внутренней точки. Методы линейного поиска и доверительных областей.

4. Метод чистого случайного поиска и его свойства. Метод роя частиц и свойства его сходимости. Симплекс-метод Нелдера – Мида и теоремы о сходимости метода. Байесовская оптимизация.

5. Теоремы существования и единственности решений систем обыкновенных дифференциальных уравнений (ОДУ). Методы Рунге – Кутты. Классический метод Рунге – Кутты четвертого порядка. Порядок метода Рунге – Кутты и теоремы о его сходимости и скорости сходимости к решению. Вложенные методы Рунге – Кутты, методы Дормана – Принса. Многошаговые методы Адамса, их порядок, выбор стартовых значений, схема «предиктор-корректор». Теорема о сходимости методов Адамса.

6. Производные решений ОДУ по параметрам. Интегрирование с выходом на ограничение. Методы решения ОДУ с ограничениями, сведение к оптимизационной задаче, примеры из механики космического полета. Методы пристрелки и параллельной пристрелки.

7. Уравнения на функции энергии, орбитального момента и вектора Лапласа в случае возмущений. Сингулярность и неустойчивость уравнений и решений в задаче двух тел. Уравнения Шперлинга – Боде. Уравнения Кустаанхеймо – Штифеля. Сравнение методов регуляризации. Уравнения в орбитальных элементах. Классические и равноденственные орбитальные элементы. Вырождение классических орбитальных элементов.

1. Общие обозначения и определения

1.1. Основные обозначения и определения

Всюду мы будем иметь дело с векторами и матрицами. Векторы будем обозначать строчными буквами с наклонным полужирным шрифтом, например $\boldsymbol{x}$. Запись $\boldsymbol{x} \in \mathbb{R}^n$ означает, что вектор $\boldsymbol{x}$ принадлежит евклидову пространству $\mathbb{R}^n$ и имеет n компонент $x_1, \ldots, x_n$. По умолчанию, векторы $\boldsymbol{x} \in \mathbb{R}^n$ будем обозначать столбцами

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \ldots \\ x_n \end{bmatrix}.$$

Матрицы будем обозначать заглавными буквам наклонным полужирным шрифтом, например $\boldsymbol{A}$. Пространство матриц размера $m \times n$ с вещественными элементами будем обозначать символом $\mathbb{R}^{m\times n}$. Вообще для простоты мы будем считать, что $\mathbb{R}^{n\times 1} = \mathbb{R}^n$, то есть к векторам будем относиться как к матрицам с одним столбцом.

Транспонирование обозначают символом T, например $\boldsymbol{A}^T$. Если $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, то $\boldsymbol{A}^T \in \mathbb{R}^{n\times m}$. Если $\boldsymbol{x} \in \mathbb{R}^n$, то $\boldsymbol{x}^T \in \mathbb{R}^{1\times n}$, где $\mathbb{R}^{1\times n}$ – это пространство строк размерности n. Матрица $\boldsymbol{A}$ называется симметрической, если $\boldsymbol{A} = \boldsymbol{A}^T$.

Скалярное произведение векторов $\boldsymbol{v} \in \mathbb{R}^n$ и $\boldsymbol{w} \in \mathbb{R}^n$ есть

$$\boldsymbol{v} \cdot \boldsymbol{w} = v_1 w_1 + \cdots + v_n w_n = \boldsymbol{v}^T \boldsymbol{w} = \boldsymbol{w}^T \boldsymbol{v} \in \mathbb{R},$$

где $\boldsymbol{v} = [v_1, \ldots, v_n]^T$ и $\boldsymbol{w} = [w_1, \ldots, w_n]^T$. Векторным произведением векторов $\boldsymbol{v} \in \mathbb{R}^3$ и $\boldsymbol{w} \in \mathbb{R}^3$ называется вектор

$$\boldsymbol{v} \times \boldsymbol{w} = \begin{bmatrix} v_2 w_3 - v_3 w_2 \\ v_3 w_1 - v_1 w_3 \\ v_1 w_2 - v_2 w_1 \end{bmatrix} = -\boldsymbol{w} \times \boldsymbol{v}.$$

Векторное произведение можно рассматривать как линейное преобразование $\boldsymbol{v} \times \boldsymbol{w} = [\boldsymbol{v}\times]\boldsymbol{w}$ с матрицей

$$[\boldsymbol{v}\times] = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}.$$

Заметим, что $\boldsymbol{v} \times \boldsymbol{w} = -\boldsymbol{w} \times \boldsymbol{v} = -[\boldsymbol{w}\times]\boldsymbol{v}$.

Введем в рассмотрение три вида нормы вектора:

$$\|\boldsymbol{v}\|_1 = |v_1| + \cdots + |v_n|, \ \boldsymbol{v} = [v_1, \ldots, v_n]^T \in \mathbb{R}^n,$$

$$\|\boldsymbol{v}\|_2 = |\boldsymbol{v}| = \sqrt{v_1^2 + \cdots + v_n^2},\ \boldsymbol{v} = [v_1, \ldots, v_n]^T \in \mathbb{R}^n,$$

$$\|\boldsymbol{v}\|_\infty = \max(|v_1|, \ldots, |v_n|),\ \boldsymbol{v} = [v_1, \ldots, v_n]^T \in \mathbb{R}^n.$$

Норма матрицы $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, подчиненная векторной норме $\|\boldsymbol{x}\|_p$, есть число

$$\|\boldsymbol{A}\|_p = \sup_{\|\boldsymbol{x}\|_p \neq 0} \frac{\|\boldsymbol{A}\boldsymbol{x}\|_p}{\|\boldsymbol{x}\|_p}.$$

Известно, что

$$\|\boldsymbol{A}\|_1 = \max_{1\le j\le n} \sum_{i=1}^{m} |a_{ij}|,\ \|\boldsymbol{A}\|_\infty = \max_{1\le i\le m} \sum_{i=1}^{n} |a_{ij}|,$$

$$\|\boldsymbol{A}\|_2 = \sqrt{\lambda_{\max}(\boldsymbol{A}^T\boldsymbol{A})}.$$

В последней формуле под знаком корня стоит максимальное собственное значение матрицы $\boldsymbol{A}^T\boldsymbol{A}$.

Мы будем часто работать с вектор-функциями $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{x})$, где $\boldsymbol{f} \in \mathbb{R}^m$, $\boldsymbol{x} \in \mathbb{R}^n$, то есть с отображениями вида $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$. Градиентом скалярной дифференцируемой функции $f(\boldsymbol{x})$, $f \in \mathbb{R}$, $\boldsymbol{x} \in \mathbb{R}^n$, называется вектор-строка

$$\frac{\partial f}{\partial \boldsymbol{x}} = \left[\frac{\partial f}{\partial x_1},\ \ldots,\ \frac{\partial f}{\partial x_n}\right] \in \mathbb{R}^{1\times n}.$$

Якобианом дифференцируемой вектор-функции $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^m$, $\boldsymbol{x} \in \mathbb{R}^n$, называется матрица $\boldsymbol{J} \in \mathbb{R}^{m\times n}$ с компонентами

$$J_{ij} = \frac{\partial f_i}{\partial x_j},\ i = 1, \ldots, m,\ i = j, \ldots, n,$$

которую будем обозначать подобно градиенту скалярной функции:

$$\boldsymbol{J} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}.$$

Иногда частную производную по переменной $\boldsymbol{x}$ функции $\boldsymbol{f}$ будем обозначать $\nabla_{\boldsymbol{x}}\boldsymbol{f}$ или просто $\nabla\boldsymbol{f}$, если ясно, о производной по какой переменной идет речь.

Введем в рассмотрение матрицу вторых производных $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ скалярной дважды дифференцируемой функции $f \in \mathbb{R}$, $\boldsymbol{x} \in \mathbb{R}^n$,

$$H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j},\ i = 1, \ldots, n,\ i = j, \ldots, n.$$

Заметим, что

$$\boldsymbol{H} = \frac{\partial}{\partial \boldsymbol{x}} \left(\frac{\partial f}{\partial \boldsymbol{x}} \right)^T.$$

Для этой матрицы будем также использовать обозначение

$$\boldsymbol{H} = \frac{\partial^2 f}{\partial \boldsymbol{x} \partial \boldsymbol{x}}.$$

Единичную матрицу $n \times n$ будем обозначать как $\boldsymbol{E}_n$. Матрицу размера $m \times n$, состоящую только из нулей, будем обозначать как $\boldsymbol{O}_{m \times n}$.

Симметрическая матрица $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ называется положительно определенной, если

$$\forall \boldsymbol{x} \in \mathbb{R}^n : |\boldsymbol{x}| \neq 0 \quad \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} > 0.$$

В этом случае мы будем писать $\boldsymbol{A} > 0$. Напомним, что любая положительно определенная матрица обратима и ее обратная также является положительно определенной матрицей.

Если $\boldsymbol{v} \in \mathbb{R}^n$, то под $\boldsymbol{v} \geq 0$ будем понимать условие неотрицательности всех компонент этого вектора, то есть $v_i \geq 0$, $i = 1, \ldots, n$. Аналогично можно ввести понятие любого другого неравенства для вектора.

2. Дифференцирование функций

2.1. Производные типичных функций

При вычислении производных от вектор-функций по вектор-аргументам удобно переходить к скалярной, индексной форме, вычислять производные по привычным правилам, а затем снова возвращаться к векторным обозначениям. Переход к индексной форме удобно выполнять, пользуясь соглашением Эйнштейна – если в выражении среди множителей встречается один и тот же индекс, то считается, что по нему осуществляется суммирование и знак суммы опускается. Главное – запомнить, какой индекс нумерует столбцы, а какой – строки. Рассмотрим несколько примеров.

Найдем производную функции $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{x}$. Это вектор-функция, поэтому ее производная – это матрица. На позиции (i, j) этой матрицы стоит

$$\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}} \right)_{ij} = \frac{\partial x_i}{\partial x_j} = \delta_j^i, \text{ откуда следует, что } \boxed{\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{E}_n},$$

где δ_{ij} – символ Кронекера: $\delta_{ij} = 1$ для $i = j$ и $\delta_{ij} = 0$ для $i \neq j$.

Рассмотрим производную функции $f(\boldsymbol{x}) = \boldsymbol{a}^T\boldsymbol{x}$. Это скалярная функция, поэтому ее производная – это вектор-строка, j-я компонента которой равна

$$\left[\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{a}^T\boldsymbol{x})\right]_j = \frac{\partial(a_i x_i)}{\partial x_j} = a_i\frac{\partial x_i}{\partial x_j} = a_i\delta_{ij} = a_j,$$

откуда, возвращаясь к матричной форме,

$$\boxed{\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{a}^T\boldsymbol{x}) = \boldsymbol{a}^T.}$$

Рассмотрим производную функции $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$. Это векторная функция, поэтому ее производная – это матрица, (i,j)-я компонента которой равна

$$\left[\frac{\partial \boldsymbol{A}\boldsymbol{x}}{\partial \boldsymbol{x}}\right]_{ij} = \frac{\partial(A_{ik}x_k)}{\partial x_j} = A_{ik}\frac{\partial x_k}{\partial x_j} = A_{ij},$$

откуда, возвращаясь к матричной форме,

$$\boxed{\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{A}\boldsymbol{x}) = \boldsymbol{A}.}$$

Рассмотрим производную функции $f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{x}$. Это скалярная функция, поэтому ее производная – это вектор-строка, j-я компонента которой равна

$$\left[\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x}^T\boldsymbol{x})\right]_j = \frac{\partial(x_i x_i)}{\partial x_j} = 2x_i\frac{\partial x_i}{\partial x_j} = 2x_i\delta_{ij} = 2x_j,$$

откуда, возвращаясь к матричной форме,

$$\boxed{\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x}^T\boldsymbol{x}) = 2\boldsymbol{x}^T}.$$

Вычислим производные еще нескольких распространненых функций:

$$\frac{\partial|\boldsymbol{x}|}{\partial \boldsymbol{x}} \to \frac{\partial\sqrt{x_i x_i}}{\partial x_j} = 2x_j\cdot\frac{1}{2\sqrt{x_i x_i}} \to \boxed{\frac{\partial|\boldsymbol{x}|}{\partial \boldsymbol{x}} = \frac{\boldsymbol{x}^T}{|\boldsymbol{x}|}.}$$

$$\frac{\partial}{\partial \boldsymbol{x}}\frac{1}{|\boldsymbol{x}|^3} \to \frac{\partial|\boldsymbol{x}|^{-3}}{\partial x_j} = -3|\boldsymbol{x}|^{-4}\cdot\frac{\partial|\boldsymbol{x}|}{\partial x_j} = -3|\boldsymbol{x}|^{-4}\cdot\frac{x_j}{|\boldsymbol{x}|} \to$$

$$\boxed{\frac{\partial}{\partial \boldsymbol{x}}\frac{1}{|\boldsymbol{x}|^3} = -3\frac{\boldsymbol{x}^T}{|\boldsymbol{x}|^5}.}$$

$$\frac{\partial}{\partial \boldsymbol{x}}\frac{\boldsymbol{x}}{|\boldsymbol{x}|^3} \to \frac{\partial}{\partial x_j}\frac{x_i}{|\boldsymbol{x}|^3} = \frac{\delta_{ij}}{|\boldsymbol{x}|^3} + x_i\frac{\partial}{\partial x_j}\frac{1}{|\boldsymbol{x}|^3} = \frac{\delta_{ij}}{|\boldsymbol{x}|^3} - \frac{3x_ix_j}{|\boldsymbol{x}|^5} \to$$

$$\boxed{\frac{\partial}{\partial \boldsymbol{x}}\frac{\boldsymbol{x}}{|\boldsymbol{x}|^3} = \frac{\boldsymbol{E}_n}{|\boldsymbol{x}|^3} - \frac{3\boldsymbol{x}\boldsymbol{x}^T}{|\boldsymbol{x}|^5}.}$$

Рассмотрим теперь уравнения задачи двух тел:

$$\begin{cases} \dot{\boldsymbol{r}} = \boldsymbol{v}, \\ \dot{\boldsymbol{v}} = -\mu\frac{\boldsymbol{r}}{|\boldsymbol{r}|^3}. \end{cases}$$

Эти уравнения можно записать в виде $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$, где

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{v} \end{bmatrix}, \ \boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{v} \\ -\mu\frac{\boldsymbol{r}}{|\boldsymbol{r}|^3} \end{bmatrix}.$$

Тогда производная правых частей уравнения по фазовому вектору

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \dfrac{\partial \boldsymbol{v}}{\partial \boldsymbol{r}} & \dfrac{\partial \boldsymbol{v}}{\partial \boldsymbol{v}} \\ \dfrac{\partial(-\mu\boldsymbol{r}/|\boldsymbol{r}|^3)}{\partial \boldsymbol{r}} & \dfrac{\partial(-\mu\boldsymbol{r}/|\boldsymbol{r}|^3)}{\partial \boldsymbol{v}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{O}_{3\times 3} & \boldsymbol{E}_3 \\ \mu\left(\dfrac{3\boldsymbol{r}\boldsymbol{r}^T}{r^5} - \dfrac{\boldsymbol{E}_3}{r^3}\right) & \boldsymbol{O}_{3\times 3} \end{bmatrix}.$$

Наконец, приведем производную функции $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$:

$$\boxed{\frac{\partial \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{x}^T\left(\boldsymbol{A} + \boldsymbol{A}^T\right).}$$

2.2. Численное дифференцирование

Рассмотрим функцию $f = f(x)$ и две схемы оценки производной этой функции в точке x. По формуле Тейлора схема

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} = f'(x) + O(h), \ h \to 0,$$

обладает первым порядком точности, если функция дважды дифференцируема в точке x. Схема

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} = f'(x) + O(h^2), \ h \to 0,$$

обладает вторым порядком точности, если функция трижды дифференцируема в точке x. Если учесть, что значение $f(x)$ уже вычислено, то первая схема требует одно дополнительное вычисление функции – в точке $x+h$. Вторая схема требует два дополнительных вычисления функции – в точке $x+h$ и точке $x-h$. Первая схема называется *правой разностной схемой* (forward difference), а вторая схема называется *центральной разностной схемой* (central difference). Можно еще вести *левую разностную схему*:

$$f'(x) \approx \frac{f(x) - f(x-h)}{h} = f'(x) + O(h),\ h \to 0,$$

которая имеет те же свойства, что и правая разностная схема, но может быть удобна, если функция не определена в точке $x+h$ ни для каких h.

На практике следует обусловиться с выбором шага h. Если этот шаг будет недостаточно малым, то существенной будет ошибка метода (ошибка аппроксимации). Если же шаг будет слишком малым, на расчетах скажутся ошибки округления и производная будет вычислена неверно. В качестве «оптимального»[1] значения h для правой и левой разностных схем можно взять $h = \sqrt{\varepsilon}x$, где ε – машинный ноль, а $x \neq 0$. Если $x = 0$, то вопрос точности можно численно исследовать отдельно, а начать можно с $h = \sqrt{\varepsilon}$. Для центральной разностной схемы оптимальным шагом обычно считается $\varepsilon^{1/3}$, безотносительно x.

Кроме того, вместо деления на h рекомендуется делить на $(x + h) - x$, с учетом ошибок округления это не одно и то же[2].

Если требуется вычислить матрицу производных $\partial \boldsymbol{f}/\partial \boldsymbol{x}$ функции $\boldsymbol{f}(\boldsymbol{x})$, то по отдельности варьируют каждую компоненту $\boldsymbol{x}$ и получают производную вектора $\boldsymbol{x}$. А именно, матрицу представляют в виде

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \left[\frac{\partial \boldsymbol{f}}{\partial x_1}\ \frac{\partial \boldsymbol{f}}{\partial x_2}\ \cdots \frac{\partial \boldsymbol{f}}{\partial x_n}\right]$$

и рассчитывают производные, например, по правой разностной схеме:

$$\frac{\partial \boldsymbol{f}}{\partial x_1} \approx \frac{\boldsymbol{f}(x_1 + h_1, x_2, \ldots, x_n) - \boldsymbol{f}(x_1, x_2, \ldots, x_n)}{h_1},$$

$$\frac{\partial \boldsymbol{f}}{\partial x_2} \approx \frac{\boldsymbol{f}(x_1, x_2 + h_2, x_3, \ldots, x_n) - \boldsymbol{f}(x_1, x_2, \ldots, x_n)}{h_2}$$

[1] См., например, Press W.H. [et al]. Numerical recipes 3rd edition: The art of scientific computing. Cambridge university press, 2007. P. 229.

[2] Подробнее о машинной арифметике можно прочитать, например, здесь: Stoer J., Bulirsch R. Introduction to Numerical Analysis. Springer-Verlag New York, 2002.

и так далее. Для правой разностной схемы придется сделать n дополнительных обращений к функции $\boldsymbol{f}$ – столько, сколько переменных. Для центральной разностной схемы придется сделать $2n$ дополнительных обращений к функции.

Для расчета производных на языке Python можно воспользоваться библиотеками `findiff`[3] и `numdifftools`[4].

3. Методы решения алгебраических уравнений

3.1. Методы решения систем линейных уравнений

Рассмотрим линейную систему уравнений вида

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{1}$$

где $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, причем $\boldsymbol{A}$ и $\boldsymbol{b}$ не зависят от $\boldsymbol{x}$. Рассмотрим несколько случаев соотношения m и n.

1) $m = n$, $\exists \boldsymbol{A}^{-1}$. В этом случае решение уравнения (1) существует, единственно и определяется по формуле

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}.$$

Численные методы решения этого уравнения обычно разделяются на прямые (точные) и приближенные методы. Прямые методы за конечное число шагов выдают точное решение задачи (если не считать ошибки округления). К прямым методам относят, например, метод Гаусса и метод прогонки (для разреженных трехдиагональных матриц). Приближенные методы представляют собой итерационные методы. Ни на какой итерации в таких методах мы не получим решение задачи, хотя мы получим приближенное решение задачи. Приближенные методы можно построить на основе метода прямой итерации, когда уравнение $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ переписывается в эквивалентном виде $\boldsymbol{x} = \boldsymbol{B}\boldsymbol{x} + \boldsymbol{c}$, а итерации удовлетворяют уравнению $\boldsymbol{x}_{k+1} = \boldsymbol{B}\boldsymbol{x}_k + \boldsymbol{c}$. К таким методам относят, например, схему $\boldsymbol{x}_{k+1} = (\boldsymbol{E}_n - \tau\boldsymbol{A})\boldsymbol{x}_k + \tau\boldsymbol{b}$, которая в случае невырожденной $\boldsymbol{A}$ сходится при любом начальном приближении $\boldsymbol{x}_0$ тогда и только тогда, когда все собственные числа матрицы $\boldsymbol{E}_n - \tau\boldsymbol{A}$ по модулю меньше единицы. Достаточное условие сходимости – неравенство $\|\boldsymbol{E}_n - \tau\boldsymbol{A}\| < 1$ в смысле какой-либо нормы. Известно, что если

[3] https://github.com/maroba/findiff.
[4] https://github.com/pbrod/numdifftools.

$\boldsymbol{x}_{k+1} = \boldsymbol{B}\boldsymbol{x}_k + \boldsymbol{c}$, $\boldsymbol{x}^* = \boldsymbol{B}\boldsymbol{x}^* + \boldsymbol{c}$ и $\|\boldsymbol{B}\| < 1$, то

$$\|\boldsymbol{x}^* - \boldsymbol{x}_k\| \leq \frac{\|\boldsymbol{B}\|}{1 - \|\boldsymbol{B}\|}\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|,$$

и таким образом можно контролировать точность определения точного решения через контроль шага итерации.

2) $m > n$, нет линейно зависимых уравнений, $\exists(\boldsymbol{A}^T\boldsymbol{A})^{-1}$. В этом случае число уравнений превышает число неизвестных, и если среди уравнений нет линейно зависимых, то решение не существует. В таких случаях мы будем искать решение, которое минимизирует невязку между левой и правой частями уравнения. А именно, мы поставим задачу безусловной оптимизации:

$$F(\boldsymbol{x}) = (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \to \min.$$

Найдем решение этой задачи. Для этого сначала упростим функцию

$$F(\boldsymbol{x}) = (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) = \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{b}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^2$$

и приравняем к нулю ее градиент

$$\frac{\partial F}{\partial \boldsymbol{x}} = 2\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A} - 2\boldsymbol{b}^T\boldsymbol{A} = 0,$$

откуда получим

$$\boldsymbol{x} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}.$$

Это – стационарная точка функции $F(\boldsymbol{x})$. Матрица $\boldsymbol{A}^T\boldsymbol{A}$ при любых условиях является неотрицательно определенной, а если она не вырождена, то является положительно определенной, а это достаточное условие оптимальности.

Технически решение оптимизационной задачи находится как решение системы линейных уравнений

$$(\boldsymbol{A}^T\boldsymbol{A})\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b},$$

в котором число уравнений и неизвестных совпадает.

3) $m < n$, переменные линейно независимы, $\exists(\boldsymbol{A}\boldsymbol{A}^T)^{-1}$. В этом случае число переменных превышает число уравнений, а в случае линейной независимости переменных это означает существование бесконечного множества решений. В этом случае нас будет интересовать решение с минимальной нормой, поэтому поставим задачу условной оптимизации:

$$\begin{cases} \boldsymbol{x}^T\boldsymbol{x} \to \min, \\ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{cases} \tag{2}$$

Найдем решение этой задачи. Составим функцию Лагранжа:

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{x}^T\boldsymbol{x} + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$$

и приравняем градиенты этой функции по $\boldsymbol{x}$ и $\boldsymbol{\lambda}$ к нулю:

$$\frac{\partial L}{\partial \boldsymbol{x}} = 2\boldsymbol{x}^T + \boldsymbol{\lambda}^T\boldsymbol{A} = 0,$$

$$\frac{\partial L}{\partial \boldsymbol{\lambda}} = (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^T = 0.$$

Из первого уравнения находим $\boldsymbol{x} = -1/2\boldsymbol{A}^T\boldsymbol{\lambda}$, подставляем во второе уравнение и находим $\boldsymbol{\lambda} = -(\boldsymbol{A}\boldsymbol{A}^T)^{-1}2\boldsymbol{b}$. Отсюда получаем

$$\boldsymbol{x} = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1}\boldsymbol{b}.$$

Так как матрица вторых производных по $\boldsymbol{x}$ функции $L(\boldsymbol{x}, \boldsymbol{\lambda})$ равна $2\boldsymbol{E}_n$ – положительно определенной матрице, то согласно достаточному условию оптимальности найденная точка $\boldsymbol{x} = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1}\boldsymbol{b}$ является точкой локального минимума, а так как задача выпуклая, то и глобального минимума (2).

Технически можно сначала численно найти решение уравнения

$$(\boldsymbol{A}\boldsymbol{A}^T)\boldsymbol{y} = \boldsymbol{b}$$

с квадратной невырожденной матрицей перед $\boldsymbol{y}$, а затем подставить полученное решение в выражение $\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{y}$.

Отмечу, что если $m > n$, то матрица $\boldsymbol{A}\boldsymbol{A}^T$ является вырожденной для любых $\boldsymbol{A} \in \mathbb{R}^{m\times n}$. Действительно, матрица $\boldsymbol{A}\boldsymbol{A}^T$ имеет размер $m \times m$, а так как

$$\text{rank}\boldsymbol{A}\boldsymbol{A}^T \leq \text{rank}\boldsymbol{A} \leq n,$$

то в случае $m > n$ получается

$$\text{rank}\boldsymbol{A}\boldsymbol{A}^T < m.$$

Таким образом, $\boldsymbol{A}\boldsymbol{A}^T$ – это матрица $m \times m$ с рангом строго меньше n, поэтому она вырождена. Это значит, что в случае $m > n$ не удастся воспользоваться формулой из пункта 3. Аналогично получается, что если $m < n$, то матрица $\boldsymbol{A}^T\boldsymbol{A}$ является вырожденной для любых $\boldsymbol{A} \in \mathbb{R}^{m\times n}$. Поэтому для этого случая решением из пункта 2 воспользоваться не удастся.

Матрицы, которые стоят в выражениях выше перед $\boldsymbol{b}$, представляют собой *псевдообратные матрицы Мура – Пенроуза* – обобщения

обратной матрицы на случай неквадратных матриц. Эти выражения для случаев $m < n$ и $m > n$ различаются, но они оба удовлетворяют условиям обычной обратной матрицы – их умножение слева и справа на обращаемую матрицу дает единичную матрицу.

Для решения систем линейных уравнений на языке Python можно воспользоваться функцией `scipy.linalg.solve` библиотеки `scipy`[5]. В MATLAB системы линейных уравнений можно решать с помощью функции `linsolve`[6].

3.2. Метод дихотомии решения уравнений

Рассмотрим уравнение вида

$$f(x) = 0$$

на отрезке $x \in [a, b]$ с непрерывной на этом отрезке функцией $f(x)$ и удовлетворяющей условию $f(a)f(b) < 0$. Это условие просто означает, что в точках a и b функция имеет разные знаки. В этом случае по теореме о среднем следует, что решение уравнения существует, хотя может быть не единственно. *Метод дихотомии* (он же *метод бисекции*, *метод половинного деления*) для решения этого уравнения при указанных условиях состоит в том, чтобы вычислить значение функции в средней точке $c = (a + b)/2$ и сравнить знак $f(c)$ со знаками $f(a)$ и $f(b)$. Если $f(a)f(c) > 0$, то рассмотрим уравнение $f(x) = 0$ на отрезке $[c, b]$, иначе – на отрезке $[a, c]$. В результате мы имеем то же уравнение, но на меньшем отрезке, на котором можно снова выбрать среднюю точку и выбрать отрезок с разными знаками функции на концах отрезка. Эту процедуру можно продолжать, пока не найдется точка (край или середина отрезка), в которой $|f(x)| < \varepsilon_f$ или пока длина отрезка не станет слишком мала. Если требуется определить корень уравнения с точностью ε, то итерации можно продолжать до тех пор, пока длина отрезка не станет меньше 2ε, тогда средняя точка в последнем отрезке будет отстоять от корня не более чем на ε.

Метод дихотомии прост в использовании и подходит для недифференцируемых функций. За одну итерацию точность определения корня увеличивается примерно вдвое, уточнение трех цифр требует 10 итераций. Если существует несколько корней уравнения, заранее не известно, к какому из них сойдется метод, хотя к одному из них он сойдется.

[5] https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.solve.html.

[6] https://www.mathworks.com/help/matlab/ref/linsolve.html.

Метод дихотомии реализован в функции `scipy.optimize.bisect` библиотеки `scipy` на Python[7].

3.3. Метод простых итераций

Рассмотрим уравнение вида $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x})$ с непрерывно дифференцируемой функцией. Зададимся начальным приближением $\boldsymbol{x}_0$ и вычислим дальнейшие приближения по формуле

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k),\ k = 0, 1, 2, \ldots$$

Если эта последовательность имеет предел, то этот предел является решением исходного уравнения. В каком случае вообще существует решение исходного уравнения? Например, если отображение $\boldsymbol{f}(\boldsymbol{x})$ является сжимающим, то есть если найдется $q < 1$, такое, что для всех $\boldsymbol{x}$, $\boldsymbol{y}$ из области определения функции $\boldsymbol{f}$ выполняется

$$|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y})| \leq q \|\boldsymbol{x} - \boldsymbol{y}\|.$$

Тогда по теореме Банаха о неподвижной точке существует неподвижная точка отображения $\boldsymbol{f}(\boldsymbol{x})$, то есть точка $\boldsymbol{x}^*$, для которой $\boldsymbol{x}^* = \boldsymbol{f}(\boldsymbol{x}^*)$. В этом случае

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| = \|\boldsymbol{f}(\boldsymbol{x}_k) - \boldsymbol{f}(\boldsymbol{x}^*)\| \leq q \|\boldsymbol{x}_k - \boldsymbol{x}^*\| \leq q^{k+1} \|\boldsymbol{x}_0 - \boldsymbol{x}^*\|,$$

что дает оценку на скорость сходимости к решению.

Если $q < 1/2$, то скорость сходимости будет быстрее, чем у метода дихотомии. Для скорости сходимости хорошо, чтобы q было как можно меньше. Если вдруг

$$\left\| \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}^*) \right\| = 0,$$

то в некоторой окрестности решения эта норма будет меньше $1/2$, в некоторой (более узкой) области норма будет меньше $1/4$ и так далее, так что оценку на q можно улучшать по мере приближения к решению, так что скорость сходимости будет выше линейной.

Описанный здесь метод решения уравнения называется *методом простых итераций*. Этот метод не накапливает ошибки вычислений, ошибки вычислений отразятся только на числе итераций, но не на точности окончательного результата. Метод простых итераций имеет важнейшие приложения для теории численных методов – на его основе

[7] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.bisect.html.

строятся метод Ньютона с квадратичной скоростью сходимости и метод Галлея с кубической скоростью сходимости, методы Рунге – Кутты можно рассматривать как методы простых итераций, теорема Пикара – Линделефа о существовании решения обыкновенных дифференциальных уравнений доказывается с использованием метода простых итераций, а также методом простых итераций решаются уравнения оптимальности в динамическом программировании и машинном обучении с подкреплением.

3.4. Метод Ньютона решения уравнений

Рассмотрим уравнение вида $\boldsymbol{f}(\boldsymbol{x}) = 0$, где $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{f} \in \mathbb{R}^n$. Зададимся некоторым начальным приближением $\boldsymbol{x}_0$, запишем линейную модель функции в точке $\boldsymbol{x}_0$ и приравняем ее к нулю:

$$\boldsymbol{l}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}_0) + \frac{\partial \boldsymbol{f}(\boldsymbol{x}_0)}{\partial \boldsymbol{x}}(\boldsymbol{x} - \boldsymbol{x}_0) = 0.$$

В результате мы получаем линейное уравнение, решение которого, в предположении о невырожденности производной функции в точке $\boldsymbol{x}_0$, равно

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 - \left[\frac{\partial \boldsymbol{f}(\boldsymbol{x}_0)}{\partial \boldsymbol{x}}\right]^{-1} \boldsymbol{f}(\boldsymbol{x}_0).$$

Полученное решение не обязательно является решением исходной системы уравнений, а является лишь приближением к нему. В полученной точке $\boldsymbol{x}_1$ снова составим линейную модель функции, приравняем ее к нулю и найдем новое приближение к решению. В общем виде итерации запишутся в виде

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[\frac{\partial \boldsymbol{f}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}}\right]^{-1} \boldsymbol{f}(\boldsymbol{x}_k),\ k = 0, 1, 2, \ldots$$

Эта схема решения уравнения и называется *методом Ньютона*. На эту схему можно посмотреть как на метод простых итераций, примененный к уравнению

$$\boldsymbol{x} = \boldsymbol{x} - \left[\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right]^{-1} \boldsymbol{f}(\boldsymbol{x}) \equiv \boldsymbol{g}(\boldsymbol{x}),$$

которое равносильно исходному уравнению. Если вычислить производную функции $\boldsymbol{g}(\boldsymbol{x})$, то получится

$$\frac{\partial \boldsymbol{g}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{E}_n - \left[\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right]^{-1} \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} - \boldsymbol{M}\boldsymbol{f}(\boldsymbol{x}) = -\boldsymbol{M}\boldsymbol{f}(\boldsymbol{x}),$$

где $\boldsymbol{M}\boldsymbol{f}(\boldsymbol{x})$ – выражение, полученное в результате дифференцирования обратной матрицы производных и умножения на функцию $\boldsymbol{f}(\boldsymbol{x})$. «Матрица» $\boldsymbol{M}$ – это тензор третьего ранга, зависящий от производных первого и второго порядка функции $\boldsymbol{f}(\boldsymbol{x})$. Из этого выражения видно, что если $\boldsymbol{M}$ ограничена, то в точке решения исходного уравнения будет $\partial \boldsymbol{g}(\boldsymbol{x})/\partial \boldsymbol{x} = 0$, а в достаточно малой окрестности решения производная будет близка к нулю (так как $\boldsymbol{f}(\boldsymbol{x})$ будет близка к нулю, а выражение пропорционально $\boldsymbol{f}(\boldsymbol{x})$). Значит, в некоторой окрестности решения норма матрицы производных $\boldsymbol{g}(\boldsymbol{x})$ будет ограничена постоянной $q < 1$, а значит, отображение $\boldsymbol{g}$ будет в этой окрестности сжимающим. Значит, метод простых итераций $\boldsymbol{x}_{k+1} = \boldsymbol{g}(\boldsymbol{x}_k)$ в этой окрестности сходится и притом с более чем линейной скоростью, так как $\partial \boldsymbol{g}(\boldsymbol{x})/\partial \boldsymbol{x} = 0$.

Итак, если функция $\boldsymbol{f}(\boldsymbol{x})$ дважды непрерывно дифференцируема, матрица $\partial \boldsymbol{f}/\partial \boldsymbol{x}$ невырождена в области интереса, а тензор $\boldsymbol{M}$ ограничен, то в достаточно малой окретности, определяемой величиной $\boldsymbol{M}$, метод Ньютона сходится со скоростью выше линейной. Можно показать, что при определенных условиях[8] это будет квадратичная скорость сходимости, то есть

$$|\boldsymbol{x}_{n+1} - \boldsymbol{x}^*| \le C|\boldsymbol{x}_n - \boldsymbol{x}^*|^2,$$

что означает, что число верных знаков после запятой удваивается после каждой итерации. Например, если в какой-то момент имеются три верных знака после запятой, то на следующей итерации будет уже шесть верных знаков, а затем 12 верных знаков. Таким образом, решение получается с высокой точностью уже за несколько первых итераций.

Рассмотрим, наконец, уравнение вида $\boldsymbol{f}(\boldsymbol{x}) = 0$, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{f} \in \mathbb{R}^m$, где m и n не равны.

Пусть $m > n$ (переопределенная система), то есть число уравнений больше числа неизвестных. Тогда, применяя метод Ньютона и решая получающуюся систему линейных уравнений методом Гаусса (минимизируя квадрат невязки), мы получаем схему

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}^T(\boldsymbol{x}_k)\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)\right]^{-1}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}^T(\boldsymbol{x}_k)\boldsymbol{f}(\boldsymbol{x}_k).$$

Если же $m < n$ (недоопределенная система), то аналогично приходим

[8] См. статью Polyak B.T. Newton's method and its use in optimization // European Journal of Operational Research. 2007. V. 181, N. 3. P. 1086–1096, и теорему 1 (Канторовича) в ней.

к схеме

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}^T \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}_k) \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}^T (\boldsymbol{x}_k) \right]^{-1} \boldsymbol{f}(\boldsymbol{x}_k).$$

Эти схемы отличаются от обычной схемы для $m = n$ просто тем, что вместо обратной матрицы рассматривается псевдообратная матрица Мура – Пенроуза.

Для решения уравнений методом Ньютона на Python можно воспользоваться функцией `scipy.optimize.newton` (только для скалярных уравнений) библиотеки `scipy`[9].

3.5. Метод градиентного спуска решения уравнения

Решение уравнения $\boldsymbol{f}(\boldsymbol{x}) = 0$ равносильно поиску глобального минимума функции

$$F(\boldsymbol{x}) = \frac{1}{2} \boldsymbol{f}^T(\boldsymbol{x}) \boldsymbol{f}(\boldsymbol{x}) \to \min .$$

Если продифференцировать это выражение по $\boldsymbol{x}$, мы получим

$$\frac{\partial F}{\partial \boldsymbol{x}} = \boldsymbol{f}^T(\boldsymbol{x}) \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}).$$

Теперь, если выбирать в качестве направления движения направление, обратное градиенту функции F, мы придем к схеме

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}^T (\boldsymbol{x}_k) \boldsymbol{f}(\boldsymbol{x}_k),$$

где α_k – это положительные числа, подбираемые так, чтобы метод сходился. Этот метод называется *методом градиентного спуска*. С точностью до множителей α_k этот метод похож на метод Ньютона, но вместо обращения матрицы производных здесь они транспонируются. Заметим, что фактором, направляющим приближения к решению уравнения у метода градиентного спуска является направление против градиента квадрата $\boldsymbol{f}$, а у метода Ньютона – сжимающее отображение.

Относительно сходимости метода известно, например, следующее. Пусть функция $F(\boldsymbol{x})$ дифференцируема и ограничена снизу. Пусть выполняется условие Липшица для градиента $\partial F/\partial \boldsymbol{x}$:

$$\|\partial F(\boldsymbol{x})/\partial \boldsymbol{x} - \partial F(\boldsymbol{y})/\partial \boldsymbol{x}\| \le L \|\boldsymbol{x} - \boldsymbol{y}\| .$$

[9] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.newton.html.

Пусть $\alpha_k = \alpha = \text{const}$ и $0 < \alpha < 2/L$. Тогда $\partial F(\boldsymbol{x}_k)/\partial \boldsymbol{x} \to 0$, $k \to \infty$, $F(\boldsymbol{x}_{k+1}) < F(\boldsymbol{x}_k)$ при любом выборе начального приближения. Таким образом, в общем случае имеется сходимость к стационарной точке функции, которая может и не быть минимумом, но быть седлом. Но если дополнительно известно, что функция F выпукла, то сходимость будет именно к минимуму, причем

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*) \le \frac{\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2}{2\alpha k},$$

где $\boldsymbol{x}^*$ – решение. Это значит, что скорость сходимости будет иметь порядок $O(1/k)$, $k \to \infty$.

Метод градиентного спуска может применяться к случаям, когда размерности $\boldsymbol{f}$ и $\boldsymbol{x}$ не совпадают. Сходимость этого метода медленнее, чем у метода Ньютона, но он может быть использован для того, чтобы получить начальное приближение для метода Ньютона. Одной из неприятностей здесь может быть то, что метод градиентного спуска может сойтись к локальному минимуму функции F, а не к глобальному минимуму, в котором F равен нулю.

3.6. Метод Левенберга – Марквардта

Методом Левенберга – Марквардта называется метод, являющийся модификацией метода Ньютона для переопределенных систем (случай $m \ge n$) и работающий по схеме:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}^T(\boldsymbol{x}_k)\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}_k) + \lambda_k \boldsymbol{E}_n\right]^{-1} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}^T(\boldsymbol{x}_k)\boldsymbol{f}(\boldsymbol{x}_k),$$

где λ_k – управляемый на каждой итерации положительный параметр. Управление этим параметром осуществляется следующим образом.

Обозначим $F(\boldsymbol{x}) = 1/2\boldsymbol{f}^T(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x})$. Если для некоторого λ_k оказалось, что $F(\boldsymbol{x}_{k+1}) < F(\boldsymbol{x}_k)$, то шаг принимается, а λ_k уменьшается, например, в 10 раз. Если же оказалось, что $F(\boldsymbol{x}_{k+1}) \ge F(\boldsymbol{x}_k)$, то шаг не принимается, а λ_k увеличивается (например, в 10 раз), и расчет $\boldsymbol{x}_{k+1}$ повторяется уже при новом значении λ_k. Процедура увеличения λ_k выполняется до тех пор, пока функция F не уменьшится.

Отметим, что при $\lambda \to \infty$:

$$-\left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}^T(\boldsymbol{x}_k)\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}_k) + \lambda_k \boldsymbol{E}_n\right]^{-1} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}^T(\boldsymbol{x}_k)\boldsymbol{f}(\boldsymbol{x}_k) \approx -\frac{1}{\lambda_k}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}^T(\boldsymbol{x}_k)\boldsymbol{f}(\boldsymbol{x}_k),$$

то есть $\boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ направлен против градиента функции F, то есть в сторону ее убывания. Поэтому если производная F не равна нулю (это можно назвать *регулярным* случаем), то при достаточно большом λ_k функция F обязана уменьшиться. Впрочем, это не означает, что большие значения λ_k эффективны: хоть направление и будет примерно совпадать с направлением антиградиента F, величина этого шага будет мала (λ_k находится в знаменателе). Поэтому по возможности λ_k следует держать как можно меньшим числом.

Направление сдвига $\boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ в методе Левенберга – Марквардта – это некоторое промежуточное направление между направлением по методу Ньютона и направлением по методу градиентного спуска. Если λ_k уменьшается, направление стремится к направлению по методу Ньютона. Если же λ_k увеличивается, направление стремится к направлению по методу градиентного спуска. Это помогает, когда начальное приближение выбрано далеко от решения уравнения и метод Ньютона не сходится. В таком случае λ_k будет увеличен, шаг будет сделан в сторону убывания нормы $\boldsymbol{f}$ и, как ожидается, в сторону окрестности, в которой метод Ньютона сойдется. В этой окрестности λ_k уменьшится, шаг будет примерно соответствовать шагу метода Ньютона и мы получим быструю сходимость к решению.

В заключение отметим, что подход с введением λ_k можно распространить и на случай недоопределенных систем:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}^T \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}_k) \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}^T (\boldsymbol{x}_k) + \lambda_k \boldsymbol{E}_m \right]^{-1} \boldsymbol{f}(\boldsymbol{x}_k).$$

Логика управления параметром λ_k остается прежней.

На Python метод Левенберга – Марквардта реализован в библиотеке `scipy` в функции `scipy.optimize.root`[10] с опцией `method='lm'`. В MATLAB этот метод реализован в функции `fsolve`[11] тулбокса `Optimization Toolbox` с опцией `Algorithm='levenberg-marquardt'`.

4. Градиентные методы оптимизации

4.1. Методы оптимизации

Методами оптимизации называются методы решения задач вида

$$f(x) \to \min,\ x \in X,$$

[10] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.root.html.

[11] https://www.mathworks.com/help/optim/ug/fsolve.html.

где $f(x)$ называется *целевой функцией*, X – *допустимым множеством*, а любой элемент $x \in X$ – *допустимым элементом*. В такой постановке объект x не обязательно является вектором и может иметь произвольную природу. Мы рассмотрим методы оптимизации функций $f(\boldsymbol{x})$, принимающих на вход векторы или конечное число переменных – так называемые задачи *математического программирования*.

Методы оптимизации условно можно разделить на градиентные и безградиентные методы. *Градиентные методы оптимизации* – это методы, использующие градиенты целевой функции или функций, описывающих допустимое множество. *Безградиентные методы* не используют градиенты этих функций.

В этом разделе расмотрим некоторые градиентные методы оптимизации, широко применяемые в механике космического полета, – методы последовательного квадратичного программирования и внутренней точки. Для этого полезно будет вспомнить необходимые и достаточные условия оптимальности в задаче математического программирования.

4.2. Условия оптимальности

Рассмотрим общую постановку задачи математического программирования:

$$\begin{cases} f(\boldsymbol{x}) \to \min, \\ \boldsymbol{g}(\boldsymbol{x}) = 0, \\ \boldsymbol{h}(\boldsymbol{x}) \le 0, \end{cases} \tag{3}$$

где $\boldsymbol{x} \in P \subset \mathbb{R}^n$, $f(\boldsymbol{x}) \in \mathbb{R}$, $\boldsymbol{g}(\boldsymbol{x}) \in \mathbb{R}^m$, $\boldsymbol{h}(\boldsymbol{x}) \in \mathbb{R}^r$. Введем функцию Лагранжа:

$$L(\boldsymbol{x}, \lambda_0, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \lambda_0 f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{\mu}^T \boldsymbol{h}(\boldsymbol{x}). \tag{4}$$

Теорема 1 (необходимые условия оптимальности). *Пусть в задаче* (3) *множество* P *выпукло, функции* f, $g_1, \ldots, g_m$, $h_1, \ldots, h_r$ *дифференцируемы в точке* $\boldsymbol{x}^* \in \operatorname{int} P$[12], *функции* $g_1, \ldots, g_m$ *непрерывно дифференцируемы в некоторой окрестности* $\boldsymbol{x}^*$. *Если* $\boldsymbol{x}^*$ – *локальное решение задачи* (3), *то существуют число* $\lambda_0^* \ge 0$ *и векторы* $\boldsymbol{\lambda}^*$ *и* $\boldsymbol{\mu}^* \ge 0$, $|\lambda_0^*| + |\boldsymbol{\lambda}^*| + |\boldsymbol{\mu}^*| > 0$, *такие, что*

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}^*, \lambda_0^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = 0, \ \mu_j^* h_j(\boldsymbol{x}^*) = 0, \ j = 1, \ldots, r.$$

Если градиенты $\nabla g_1(\boldsymbol{x}^*)$, $\ldots$, $\nabla g_m(\boldsymbol{x}^*)$, $\nabla h_1(\boldsymbol{x}^*)$, $\ldots$, $\nabla h_r(\boldsymbol{x}^*)$ *линейно независимы, то* $\lambda_0^* \ne 0$.

[12]Символ $\operatorname{int} P$ обозначает множество внутренних точек множества P.

Введем в рассмотрение множество

$$\mathcal{I}(\boldsymbol{x}) = \{i = 1, \ldots, r : h_i(\boldsymbol{x}) = 0\},$$

которое содержит номера компонент функций ограничений-неравенств, которые в точке $\boldsymbol{x}$ обращаются в нуль. Соответствующие ограничения-неравенства называются *активными*, а остальные неравенства – *пассивными*. Введем также множество $H(\boldsymbol{x})$ всех векторов $\boldsymbol{v} \in \mathbb{R}^n$, таких, что

$$\nabla f(\boldsymbol{x})\boldsymbol{v} \le 0,\ \nabla g_i(\boldsymbol{x})\boldsymbol{v} = 0,\ \forall i = 1, \ldots, m,\ \nabla h_j(\boldsymbol{x})\boldsymbol{v} \le 0,\ \forall j \in \mathcal{I}(\boldsymbol{x}).$$

Геометрически это множество векторов, вдоль которых локально целевая функция не возрастает, а ограничения-равенства и ограничения-неравенства не нарушаются.

Теорема 2 (достаточные условия оптимальности). *Пусть в задаче* (3) *функции* f, g_1, …, g_m, h_1, …, h_r *дважды дифференцируемы в точке* $\boldsymbol{x}^* \in \operatorname{int} P$. *Предположим, что существуют число* $\lambda_0^* \ge 0$ *и векторы* $\boldsymbol{\lambda}^*$ *и* $\boldsymbol{\mu}^* \ge 0$ *такие, что выполняются условия*

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}^*, \lambda_0^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = 0,\ \mu_j^* h_j(\boldsymbol{x}^*) = 0,\ i = j, \ldots, r,$$

$$\boldsymbol{v}^T \frac{\partial^2 L}{\partial \boldsymbol{x} \partial \boldsymbol{x}} \boldsymbol{v} > 0,\ \forall \boldsymbol{v} \in H(\boldsymbol{x}^*).$$

Тогда $\boldsymbol{x}^*$ *– строгое локальное решение задачи* (3), *то есть* $f(\boldsymbol{x}^*) < f(\boldsymbol{x})$ *при всех* $\boldsymbol{x} \in \mathbb{R}^n$, *достаточно близких к* $\boldsymbol{x}^*$, *но отличных от* $\boldsymbol{x}^*$.

Описанные в теореме 1 условия

$$\lambda_0^* \ge 0, \tag{5}$$

$$\boldsymbol{\mu}^* \ge 0, \tag{6}$$

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}^*, \lambda_0^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = 0, \tag{7}$$

$$\mu_j^* h_j(\boldsymbol{x}^*) = 0,\ j = 1, \ldots, r, \tag{8}$$

$$\boldsymbol{g}(\boldsymbol{x}^*) = 0, \tag{9}$$

$$\boldsymbol{h}(\boldsymbol{x}^*) \le 0 \tag{10}$$

представляют собой необходимые условия оптимальности и называются *условиями Каруша – Куна – Таккера.*

Заметим, что если градиенты $\nabla g_1(\boldsymbol{x})$, …, $\nabla g_m(\boldsymbol{x})$, $\nabla h_1(\boldsymbol{x})$, …, $\nabla h_r(\boldsymbol{x})$ линейно независимы в каждой точке $\boldsymbol{x} \in \mathbb{R}^n$, то они линейно независимы в том числе и в точке оптимальности, поэтому $\lambda_0 \ne 0$ и всегда можно считать $\lambda_0 = 1$. В таких случаях достаточно ограничиться функцией Лагранжа вида

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g} + \boldsymbol{\mu}^T \boldsymbol{h}.$$

4.3. Методы последовательного квадратичного программирования

Методы последовательного квадратичного программирования (*sequential quadratic programming, SQP*) – это одни из самых эффективных численных методов решения задач нелинейного программирования вида (3).

Чтобы понять суть этих методов, вспомним метод Ньютона. Пусть нам дано уравнение $\boldsymbol{f}(\boldsymbol{x}) = 0$ с нелинейной функцией $\boldsymbol{f}(\boldsymbol{x})$. Мы умеем решать линейные уравнения, как же решить нелинейное? Очень просто – аппроксимировать в данной точке $\boldsymbol{x}_0$ функцию $\boldsymbol{f}(\boldsymbol{x})$ ее линейной частью $\boldsymbol{l}_0(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}_0) + \boldsymbol{A}_0(\boldsymbol{x} - \boldsymbol{x}_0)$ и решить линейное уравнение $\boldsymbol{l}_0(\boldsymbol{x}) = 0$. Оно имеет решение $\boldsymbol{x}_1$, которое (в силу нелинейности $\boldsymbol{f}(\boldsymbol{x})$), скорее всего, не является корнем функции $\boldsymbol{f}(\boldsymbol{x})$. Поэтому мы строим новую линейную функцию $\boldsymbol{l}_1(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}_1) + \boldsymbol{A}_1(\boldsymbol{x} - \boldsymbol{x}_1)$ и находим решение $\boldsymbol{x}_2$ уравнения $\boldsymbol{l}_1(\boldsymbol{x}) = 0$. Получается, что решение нелинейного уравнения $\boldsymbol{f}(\boldsymbol{x})$ методом Ньютона – это процедура последовательного решения ряда линейных систем уравнений.

Вернемся к задаче нелинейного программирования (3). Оказывается, что существуют эффективные алгоритмы решения задач квадратичного программирования, то есть задач с квадратичной целевой функцией и линейными ограничениями. Поэтому предлагается следующая схема: вместо решения исходной задачи на каждой итерации решать некоторую задачу квадратичного программирования. Необходимо лишь обеспечить сходимость последовательности $\boldsymbol{x}_0$, $\boldsymbol{x}_1$, ... решений квадратичных задач к решению $\boldsymbol{x}^*$ исходной задачи. Этот метод решения исходной задачи нелинейного программирования и называется *методом последовательного квадратичного программирования*, или проще – *методом SQP*.

Метод SQP – это не какой-то один метод, а класс методов. Методы внутри этого класса отличаются постановкой и методами решения квадратичной подзадачи, методами оперирования с ограничениями-неравенствами и методами выбора шага.

Перейдем к чуть более детальному описанию метода SQP. Пусть $\boldsymbol{x}_k$ – приближение к решению $\boldsymbol{x}^*$ исходной задачи. В качестве квадратичной подзадачи можно рассмотреть задачу

$$\frac{1}{2}\Delta\boldsymbol{x}_k^T \boldsymbol{H}_k \Delta\boldsymbol{x}_k + \nabla f(\boldsymbol{x}_k)\Delta\boldsymbol{x}_k \to \min$$

при ограничениях

$$\boldsymbol{g}(\boldsymbol{x}_k) + \nabla\boldsymbol{g}(\boldsymbol{x}_k)\Delta\boldsymbol{x}_k = 0,$$
$$\boldsymbol{h}(\boldsymbol{x}_k) + \nabla\boldsymbol{h}(\boldsymbol{x}_k)\Delta\boldsymbol{x}_k \le 0,$$

где $\boldsymbol{H}_k = \nabla^2_{\boldsymbol{x}} L(\boldsymbol{x}_k)$ – матрица вторых производных функции Лагранжа.

Обращаю внимание на то, что на месте $\boldsymbol{H}_k$ стоит матрица вторых производных именно функции Лагранжа, а не целевой функции f, что было бы логично, раз уж строится квадратичная модель исходной задачи с минимизацией функции f. Но в использовании в качестве $\boldsymbol{H}_k$ матрицы вторых производных функции Лагранжа есть причина. Оказывается[13], что в случае только ограничений-равенств если взять условия Каруша – Куна – Таккера для исходной задачи оптимизации и решать их методом Ньютона, то полученная схема будет в точности совпадать с условиями Каруша – Куна – Таккера для написанной выше квадратичной задачи с $\boldsymbol{H}_k$, равной матрице вторых производных именно функции Лагранжа. Это значит, что условия сходимости и факты о скорости сходимости метода Ньютона переносятся на квадратичную подзадачу.

Еще стоит заметить, что в силу линейного ограничения-равенства в квадратичной подзадаче она эквивалентна задаче, в которой вместо ∇f стоит $\nabla_{\boldsymbol{x}} L$. В итоге это значит, что в качестве квадратичной модели рассматривается квадратичная модель функции Лагранжа, а не целевой функции.

Так как расчет матрицы вторых производных обычно является трудоемкой операцией, были изобретены различные способы оценивания этой матрицы. Например, можно воспользоваться формулой Бройдена – Флетчера – Гольдфарба – Шанно (BFGS):

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k + \frac{\boldsymbol{q}_k \boldsymbol{q}_k^T}{\boldsymbol{q}_k^T \boldsymbol{s}_k} - \frac{\boldsymbol{H}_k \boldsymbol{s}_k \boldsymbol{s}_k^T \boldsymbol{H}_k^T}{\boldsymbol{s}_k^T \boldsymbol{H}_k \boldsymbol{s}_k},$$

где $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$, $\boldsymbol{q}_k = \nabla_{\boldsymbol{x}} L(\boldsymbol{x}_{k+1}) - \nabla_{\boldsymbol{x}} L(\boldsymbol{x}_k)$. На первой итерации можно выбрать $\boldsymbol{H}_0 = \boldsymbol{E}_n$, последующие значения $\boldsymbol{H}_k$ будут корректироваться и лучше отражать информацию о кривизне. Кроме того, на практике рекомендуется[14] делать так, чтобы матрицы $\boldsymbol{H}_k$ были положительно определенными, этого можно достичь, определенным образом скорректировав вектор $\boldsymbol{q}_k$. С одной стороны, задача квадратичного программирования имеет в этом случае единственное решение. С другой стороны, исследования показывают, что аппроксимация матрицы вторых производных положительно определенной матрицей способствует сходимости, даже если истинная матрица вторых производных не является положительно определенной (см. ту же статью M.J.D. Powell).

[13] Nocedal J., Wright S.J. Numerical Optimization. Springer, 2006. P. 531–532.

[14] Powell M.J.D. A Fast Algorithm for Nonlinearly Constrained Optimization Calculations // Numerical Analysis, G.A. Watson ed., Lecture Notes in Mathematics, Springer Verlag. 1978. Vol. 630.

Когда решаются оптимизационные задачи с ограничениями-неравенствами, для удовлетворения условий Каруша – Куна – Таккера нужно выяснять, какие из ограничений-неравенств являются активными в оптимальной точке. В рамках методов SQP на этот счет есть два подхода. В одном подходе – *inequality-constrained QP, IQP* – активными ограничениями считаются те, которые являются активными в квадратичной подзадаче. В другом подходе – *equality constrained QP, EQP* – формируется предположение о том, какие именно из ограничений являются активными, и в рамках этого предположения решается квадратичная задача только с ограничениями-равенствами и активными ограничениями. Предположение об активных ограничениях может меняться от подзадачи к подзадаче. В конце концов нужно найти такое множество активных ограничений, чтобы выполнялись условия оптимальности.

Решение квадратичной подзадачи дает приближенный шаг метода $\boldsymbol{d}_k = \Delta\boldsymbol{x}_k$. Этот шаг может увеличить значение целевой функции $f(\boldsymbol{x})$ и нарушить ограничения-равенства (9) и неравенства (10). Чтобы этого избежать, можно применить процедуру линейного поиска: после каждого решения квадратичной задачи искать значение скалярного параметра α:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha\boldsymbol{d}_k,$$

которое оптимизирует скалярную функцию

$$\Psi(\alpha) = f(\boldsymbol{x}_{k+1}) + \sum_{j=1}^{m} w_j |g_j(\boldsymbol{x}_{k+1})| + \sum_{j=1}^{r} \bar{w}_j \max(0, h_j(\boldsymbol{x}_{k+1})),$$

где w_j и $\bar{w}_j$ – положительные весовые коэффициенты, которые можно брать равными друг другу.

Приведу теорему о сходимости метода SQP в случаях, когда даны только ограничения-равенства (или когда даны ограничения-неравенства, но известны активные ограничения).

Теорема 3 (о сходимости метода SQP)[15]. *Пусть последовательности $\{\boldsymbol{x}_k\}$ и $\{\boldsymbol{x}_k + \boldsymbol{d}_k\}$, генерируемые описанным выше методом SQP, содержатся в замкнутом и ограниченном множестве, в котором функции f, g_1, …, g_m имеют непрерывные производные. Пусть матрицы $\boldsymbol{H}_k$ и множители Лагранжа $\boldsymbol{\lambda}_k$ ограничены и пусть весовые коэффициенты w_j и $\bar{w}_j$ одинаковы и равны постоянной μ, которая удовлетворяет неравенству $\mu \geq \|\boldsymbol{\lambda}_k\|_\infty + \rho$ для всех k и для некоторой постоянной $\rho > 0$. Тогда все предельные точки последовательности $\{\boldsymbol{x}_k\}$ удовлетворяют необходимым условиям оптимальности.*

[15] Nocedal J., Wright S.J. Numerical Optimization. Springer, 2006. P. 557.

В упомянутой монографии Nocedal J., Wright S.J. Numerical Optimization можно найти теоремы о скорости сходимости метода SQP. Как уже было сказано, при выполнении определенных условий на функции в задаче оптимизации и некоторых других условий скорость сходимости будет квадратичная, так как метод SQP фактически совпадает с методом Ньютона, который применяется для условий Каруша – Куна – Таккера исходной задачи. Если вместо точной матрицы вторых производных используется приближеная формула, то при определенных условиях получается сверхлинейная сходимость.

Метод последовательного квадратичного программирования реализован в функции `fmincon`[16] тулбокса `Optimization Toolbox` с опциями `Algorithm='sqp'` и `Algorithm='active-set'`.

4.4. Методы внутренней точки

Методы внутренней точки (*interior-point methods*), они же – *методы штрафных функций* (*barrier methods*), – это класс методов, которые применяются для решения оптимизационных задач с ограничениями-неравенствами и суть которых состоит в том, чтобы свести задачу оптимизации с неравенствами к серии задач без неравенств, предел решений которых дает предел решения исходной задачи. Такой подход позволяет обойти проблему поиска активных ограничений, которая появляется при использовании методов SQP. На данный момент эти методы считаются наиболее эффективными для задач большой размерности.

Пусть дана задача математического программирования (3). На ее основе рассмотрим семейство приближенных задач, параметризованное числом $\mu > 0$:

$$\begin{cases} f_\mu(\boldsymbol{x}, \boldsymbol{s}) = f(\boldsymbol{x}) - \mu \sum\limits_{i=1}^{r} \ln s_i \to \min, \\ \boldsymbol{g}(\boldsymbol{x}) = 0, \\ \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{s} = 0, \end{cases} \tag{11}$$

где $\boldsymbol{s} > 0$ – это так называемая фиктивная переменная (slack variable). Она вводится для того, чтобы избавиться от ограничений-неравенств. В формулировке оптимизационной задачи условие $\boldsymbol{s} > 0$ не требуется, так как достаточно в качестве начального приближения выбрать $\boldsymbol{s}_0 > 0$ и дальнейшие приближения все время будут оставаться положительными, так как при приближении $\boldsymbol{s}$ к нулю целевая функция неограниченно возрастает.

[16] https://www.mathworks.com/help/optim/ug/fmincon.html.

Очевидно, что при $\mu \to 0$ приближенная целевая функция $f_\mu(\boldsymbol{x}, \boldsymbol{s})$ сходится к исходной целевой функции $f(\boldsymbol{x})$. Естественно ожидать, что и решение приближенной задачи $\boldsymbol{x}^* = \boldsymbol{x}^*(\mu)$ при $\mu \to 0$ сходится к решению исходной задачи (3). Мы оставляем в стороне вопрос о том, действительно ли сходимость решений имеет место, но будем это предполагать. Известны условия, при которых такая сходимость будет иметь место[17].

Метод внутренней точки заключается в том, чтобы последовательно решать задачи (11) для некоторой убывающей к нулю последовательности μ_k, причем решение, полученное для μ_k, можно использовать в качестве начального приближения для μ_{k+1}. Решение приближенной задачи представляет собой отыскание тем или иным методом точки, удовлетворяющей условиям Каруша – Куна – Таккера, например методом Ньютона или даже методом SQP. Известны теоремы о сходимости метода внутренней точки[18].

После каждого решения приближенной задачи алгоритм может выполнять линейный поиск, оптимизируя скалярную функцию

$$\Psi(\alpha) = f_\mu(\boldsymbol{x}_{k+1}, \boldsymbol{s}_{k+1}) + \nu_k \left\|\boldsymbol{g}(\boldsymbol{x}_{k+1})\right\|_\infty + \nu_k \left\|\boldsymbol{h}(\boldsymbol{x}_{k+1}) + \boldsymbol{s}_{k+1}\right\|_\infty,$$

где $(\boldsymbol{x}_{k+1}, \boldsymbol{s}_{k+1}) = (\boldsymbol{x}_k, \boldsymbol{s}_k) + \alpha \boldsymbol{d}_k$, $\boldsymbol{d}_k$ – шаг решения приближенной задачи, а параметр $\nu_k > 0$ может меняться от шага к шагу. Увеличение этого параметра приближает искомую оптимальную точку $(\boldsymbol{x}_{k+1}, \boldsymbol{s}_{k+1})$ к множеству

$$\{(\boldsymbol{x}, \boldsymbol{s}) : \boldsymbol{g}(\boldsymbol{x}) = 0,\ \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{s} = 0\}.$$

Обратим также внимание на то, что с практической точки зрения оптимизация функции $f_\mu(\boldsymbol{x}, \boldsymbol{s})$ – это не то же самое, что оптимизация функции

$$f(\boldsymbol{x}) - \mu \sum_{i=1}^{r} \ln(-h_i(\boldsymbol{x})).$$

Дело в том, что это выражение определено лишь для $h_i(\boldsymbol{x}) < 0$ и для старта метода требуется точка $\boldsymbol{x}_0$, удовлетворяющая этому неравенству. Что касается функции $f_\mu(\boldsymbol{x}, \boldsymbol{s})$, достаточно в качестве начального приближения брать какой-либо $\boldsymbol{s}_0 > 0$, равенство $\boldsymbol{h}(\boldsymbol{x}_0) + \boldsymbol{s}_0 = 0$ при этом может быть и не выполнено, но ничего страшного в этом нет, так как итерации метода, с одной стороны, скорректируют $\boldsymbol{x}$ для равенства в $\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{s} = 0$, а с другой стороны не дадут $\boldsymbol{s}$ приблизиться к нулю.

[17] Nocedal J., Wright S.J. Numerical Optimization. Springer. 2006. P. 565.
[18] Nocedal J., Wright S.J. Numerical Optimization. Springer. 2006. P. 568.

Метод внутренней точки реализован в MATLAB в функции `fmincon`[19] с опцией `Algorithm='interior-point'`. На Python метод внутренней точки реализован в функции `scipy.optimize.minimize`[20] библиотеки `scipy` с опцией `method='trust-constr'`.

4.5. Борьба с ограничениями-неравенствами

Одна из основных проблем решения задачи нелинейного программирования состоит в том, чтобы выяснить, какие ограничения-неравенства являются активными, а какие нет. Это очень важно, потому как знание активных ограничений позволяет выкинуть остальные. Есть два основных подхода решения этой проблемы.

Первый подход – метод активных ограничений (active-set methods). На каждой итерации метода оптимизации (например, SQP) производится оценка или коррекция множества активных ограничений. А именно, сначала выбирается некоторое множество $\mathcal{W}$ ограничений-неравенств, оно называется *рабочим множеством* (*working set*). Все ограничения-неравенства не из этого множества игнорируются, а ограничения из $\mathcal{W}$ считаются активными. Тогда мы можем поставить оптимизационную задачу только с ограничениями-равенствами. В результате решения получается вектор основных переменных $\boldsymbol{x}^*$. Если для него удается подобрать вектор множителей Лагранжа $\boldsymbol{\mu}^*$, удовлетворяющий условиям Каруша – Куна – Таккера, то шаг принимается и процедура продолжается. Если же подобрать множители Лагранжа не получается, то рабочее множество $\mathcal{W}$ корректируется. Бывает так, что корректировать это множество приходится часто. Это – комбинаторная проблема метода активных ограничений. Второй подход – методы внутренней точки. Методы внутренней точки изначально формулируют исходную задачу в виде оптимизационной задачи только с ограничениями-равенствами и генерируют последовательность приближений $\boldsymbol{x}_k$ далеко от границ ограничений-неравенств.

Практика показала, что методы активных ограничений отлично подходят для задач с небольшим числом ограничений и переменных, и когда начальное приближение удовлетворяет ограничениям. Методы внутренней точки оказываются эффективными для задач с большим числом ограничений и переменных, даже когда начальное приближение не удовлетворяет ограничениям. Для них не стоит комбинаторная проблема, характерная для методов активных ограничений.

[19] https://www.mathworks.com/help/optim/ug/fmincon.html.

[20] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html.

4.6. Выбор шага метода

Ранее мы видели, что SQP и метод внутренней точки решают сначала некоторую подзадачу и находят шаг $\boldsymbol{d} = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$. Но так как подзадача является лишь приближением к исходной задаче, то точка $\boldsymbol{x}_k + \boldsymbol{d}$ может привести к увеличению целевой функции или выходу за ограничения. Другими словами, решение подзадачи не всегда адекватно отражает направление к решению исходной задачи. Для решения этой проблемы обычно применяется одна из двух стратегий:

1) Линейный поиск (line search methods): вводится параметр $\alpha > 0$, значение которого подбирается таким образом, чтобы оптимизировать в точке $\boldsymbol{x}_{k+1}(\alpha) = \boldsymbol{x}_k + \alpha\boldsymbol{d}$ какую-либо функцию качества, например, взвешенную комбинацию целевой функции и ограничений. Этот подход и был продемонстрирован на примере методов SQP и метода внутренней точки. После того, как оптимальное значение α^* найдено, в качестве следующей итерации рассматривают $\boldsymbol{x}_{k+1}(\alpha^*)$.

2) Метод доверительных областей (trust-region methods): в оптимизационную подзадачу добавляется еще одно условие: $\|\boldsymbol{d}\| \leq \Delta$. Множество $\{\boldsymbol{d} : \|\boldsymbol{d}\| \leq \Delta\}$ называется доверительной областью, а Δ – это размер доверительной области. Если $\boldsymbol{x}_{k+1}(\Delta) = \boldsymbol{x}_k + \boldsymbol{d}(\Delta)$ приводит к увеличению значения целевой функции или выходу за ограничения, то размер области Δ отклоняется, значение Δ уменьшается, и подзадача решается снова, с новым значением Δ. Если же значение целевой функции уменьшается, а ограничения не нарушаются, то Δ увеличивается.

Заметим, что в случае линейного поиска фиксируется направление шага, а подбирается его длина. В методах доверительных областей фиксируется длина шага, а подбирается его направление.

В MATLAB в упомянутой ранее функции `fmincon` алгоритмы `interior-point` и `sqp` используют линейный поиск, а `trust-region-reflective` основан на методе доверительных областей. На Python упомянутая функция `scipy.optimize.minimize` с опциями `trust-constr`, `trust-ncg`, `trust-krylov`, `trust-exact`, `dogleg` основываются на методе доверительных областей.

5. Безградиентные методы оптимизации

5.1. Метод чистого случайного поиска

Метод чистого случайного поиска (*pure random search*) – это безградиентный метод оптимизации, который применяется для решения

задач вида

$$f(\boldsymbol{x}) \to \min,\ \boldsymbol{x} \in X \subset \mathbb{R}^n.$$

Метод состоит в генерации реализаций случайной величины (или вектора) $\boldsymbol{\xi}$ с некоторым распределением на множестве X и выбором той реализации, значение функции f в которой минимально. Можно генерировать заранее заданное число реализаций, можно генерировать их до тех пор, пока не будет найдена реализация, которой соответствует достаточно малое значение функции f. В этом случае число реализаций тоже является случайной величиной.

Пусть в множестве X имеется множество U, любые значения $\boldsymbol{x}$ из которой нас устраивают с точки зрения значений функции f. Пусть вероятность попасть в это множество равна ε, то есть $\mathbb{P}(\boldsymbol{X}_i \in U) = \varepsilon$, где $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ – независимые случайные величины (или векторы), распределенные так же, как $\boldsymbol{\xi}$, то есть выборка из распределения $\boldsymbol{\xi}$. Тогда вероятность того, что ни один из $\boldsymbol{X}_i$ не попадет в множество U, равна

$$\mathbb{P}(\forall i = 1, \ldots, N\ \boldsymbol{X}_i \notin U) = (1-\varepsilon)^N.$$

Вероятность того, что хотя бы один $\boldsymbol{X}_i$ попадет в множество U, равна

$$\mathbb{P}(\exists i = 1, \ldots, N\ \boldsymbol{X}_i \in U) = 1 - (1-\varepsilon)^N.$$

Обозначим эту вероятность как γ. Тогда число испытаний N, которое требуется для попадания в множество U хотя бы одной реализации с вероятностью γ, равно

$$N = \frac{\ln(1-\gamma)}{\ln(1-\varepsilon)}.$$

Например, если $\varepsilon = 10\%$, $\gamma = 90\%$, то $N = 22$, если $\varepsilon = 5\%$, $\gamma = 95\%$, то $N = 60$, если $\varepsilon = 1\%$, $\gamma = 99\%$, то $N = 460$. Обратим внимание на то, что число испытаний не зависит от размерности пространства X.

Если $\boldsymbol{\xi}$ имеет непрерывное равномерное распределение на X, то вероятность ε попадания в множество U равна объему множества U относительно X.

5.2. Метод роя частиц

Метод роя частиц (*particle swarm optimization, PSO*) – это безградиентный метод оптимизации, изобретенный инженером Расселом Эберхартом и психологом Джеймсом Кеннеди в 1995 году, когда они моделировали движение стаи птиц вокруг источника еды и поняли, что их поведение напоминает решение оптимизационной задачи.

Метод применяется для решения задач вида

$$\begin{cases} f(\boldsymbol{x}) \to \min, \\ \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}, \end{cases}$$

где $\boldsymbol{x}$, $\boldsymbol{l}$, $\boldsymbol{u} \in \mathbb{R}^n$. Идея метода состоит в том, чтобы инициализировать в пространстве состояний группу (рой) $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ переменных (частиц) и привести их в движение в этом пространстве. Обычно число частиц N много больше размерности пространства n. В процессе движения эти частицы обмениваются между собой информацией о своих лучших с точки зрения целевой функции положениях.

Опустив некоторые детали реализации (например, условия остановки и коррекцию скорости при вылете частицы за границы рассматриваемого множества), рассмотрим простейший вариант метода.

Сначала случайным образом инициализируем в пространстве переменные $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, например в соответствии с непрерывным равномерным распределением. Значения этих переменных будут в этот момент времени лучшими положениями, найденными каждой частицей, эти лучшие положения обозначим как $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N$. Одно из этих лучших положений для каждой частицы будет лучшим из всех положений, найденных роем, обозначим его $\boldsymbol{g}$. Далее случайным образом присвоим каждой частице скорость $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ и будем рассматривать динамику частиц в дискретные моменты времени. На каждом временном шаге будем обновлять лучшие положения для каждой частицы и лучшее положение $\boldsymbol{g}$, найденное за все время движения роя. Скорость каждой частицы будем обновлять согласно следующему правилу:

$$\boldsymbol{v}_i^{(k+1)} = \boldsymbol{\omega}_v \odot \boldsymbol{v}_i^{(k)} + \boldsymbol{\omega}_p \odot \boldsymbol{\rho}_{ip}^{(k)} \odot (\boldsymbol{p}_i^{(k)} - \boldsymbol{x}_i^{(k)}) + \boldsymbol{\omega}_g \odot \boldsymbol{\rho}_{ig}^{(k)} \odot (\boldsymbol{g}^{(k)} - \boldsymbol{x}_i^{(k)}),$$

где i – номер частицы, k – номер шага по времени, $\odot$ означает поэлементное умножение векторов, $\boldsymbol{\omega}_v$, $\boldsymbol{\omega}_p$, $\boldsymbol{\omega}_g$ – векторы весовых коэффициентов, $\boldsymbol{\rho}_{ip}^{(k)}$, $\boldsymbol{\rho}_{ig}^{(k)}$ – случайные векторы с независимыми компонентами, распределенными, например, равномерно на отрезке $[0, 1]$. Согласно этой формуле скорость частицы на следующем шаге определяется частично (с некоторым весом) скоростью на текущем шаге, направлением на лучшее положение этой частицы $\boldsymbol{p}_i^{(k)}$ до момента k и направлением на лучшее положение $\boldsymbol{g}^{(k)}$, найденное роем за всю историю движения роя до момента k.

Первое слагаемое в формуле называется *инерционной* составляющей скорости частицы, второе слагаемое – *когнитивной* составляющей, а третье слагаемое – *социальной* составляющей. Весовые коэффициенты позволяют регулировать соотношение между этими направлениями,

а случайные величины перед вторым и третьим слагаемыми отвечают за случайность приоритета направлений. Чем больше весовые коэффициенты $\boldsymbol{\omega}_v$ в сравнении с $\boldsymbol{\omega}_p$ и $\boldsymbol{\omega}_g$, тем больше склонность частицы двигаться в том же направлении, что и до коррекции скорости. Аналогично, чем выше $\boldsymbol{\omega}_p$, тем больше склонность частицы двигаться по направлению к своему лучшему положению. Чем выше $\boldsymbol{\omega}_g$, тем больше склонность частицы двигаться по направлению к лучшему из всех найденных роем положению $\boldsymbol{g}$.

Движение частиц продолжается до тех пор, пока не будут выполнены критерии остановки. Например, движение может останавливаться, когда найдено положение $\boldsymbol{g}$ с достаточно малым значением функции f или когда превышено допустимое число шагов.

Бывает так, что рой «слетается» к локальному минимуму и застревает в нем. В этом случае помогает разбиение роя на группы, введение $\boldsymbol{g}_j$ – лучших положений, найденных в каждой группе, и коррекции формулы для изменения скорости – вместо $\boldsymbol{g}$ как лучшего положения среди всех частиц роя можно подставить $\boldsymbol{g}_j$ – лучшее положение среди частиц группы, которой принадлежит данная частица.

Эффективность метода существенно зависит от весовых коэффициентов. Существуют многочисленные способы управления этими параметрами. Например, в реализации метода роя частиц в функции `particleswarm`[21] библиотеки Global Optimization Toolbox в MATLAB есть счетчик c, который вычисляет, сколько раз алгоритму не удалось улучшить общее решение. Сначала инициализируется $c = 0$. Пока $c < 2$, величина $\boldsymbol{\omega}_v$ увеличивается в два раза, то есть инерция повышается. При $c \in [2, 5]$ коэффициент инерции $\boldsymbol{\omega}_v$ не изменяется, а при $c > 5$ коэффициент уменьшается в два раза. Это значит, что рой становится менее «вязким», если ему долго не удается улучшить решение. Если же решение улучшается, то c уменьшается на 1. После нескольких улучшений рой может стать более вязким, что может способствовать сходимости к решению.

Так как метод является стохастическим (в нем используются реализации случайных величин), то последовательно запуская метод, мы будем каждый раз получать новое решение. В задачах механики космического полета метод роя частиц часто применяют для получения начального приближения решения оптимизационной задачи, чтобы использовать его в градиентных методах оптимизации.

Интересно отметить, что метод роя частиц описывает линейную динамическую систему. Это наблюдение позволяет проанализировать

[21] https://www.mathworks.com/help/gads/particleswarm.html.

метод на сходимость, по крайней мере, к лучшему из найденных роем решений.

Предположим для простоты, что лучшее положение i-й частицы $\boldsymbol{p}_i$ и лучшее положение, которое нашел рой за все время $\boldsymbol{g}$, не меняются от итерации к итерации. Это предположение вполне состоятельно в случаях, когда рой долго не может найти решение. Кроме того, предположим для простоты, что случайные векторы $\boldsymbol{\rho}_{ip}^{(k)}$ и $\boldsymbol{\rho}_{ig}^{(k)}$ в формуле для скорости частицы заменены на $1/2$ – среднее значение случайной величины, равномерно распределенной на $[0, 1]$. Тогда динамика частицы описывается неслучайными и притом *линейными* уравнениями:

$$\boldsymbol{v}_i^{(k+1)} = \boldsymbol{\omega}_v \odot \boldsymbol{v}_i^{(k)} + \frac{1}{2}\boldsymbol{\omega}_p \odot (\boldsymbol{p}_i - \boldsymbol{x}_i^{(k)}) + \frac{1}{2}\boldsymbol{\omega}_g \odot (\boldsymbol{g} - \boldsymbol{x}_i^{(k)}),$$

$$\boldsymbol{x}_i^{(k+1)} = \boldsymbol{x}_i^{(k+1)} + \boldsymbol{v}_i^{(k+1)}.$$

Более того, эти уравнения покомпонентно отделяются друг от друга. Общий вид уравнения для каждой из размерности векторов $\boldsymbol{v}$ и $\boldsymbol{x}$ для каждой частицы можно записать так:

$$v_{k+1} = av_k + b(p - x_k),\ x_{k+1} = x_k + v_{k+1},$$

где введены обозначения

$$a = w_v,\ b = \frac{w_p + w_g}{2},\ p = \frac{w_p}{w_p + w_g}p_i + \frac{w_g}{w_p + w_p}g.$$

Полученные скалярные линейные уравнения для v, x можно записать в виде

$$\boldsymbol{y}_{k+1} = \boldsymbol{A}\boldsymbol{y}_k + \boldsymbol{B}p,$$

где

$$\boldsymbol{y}_k = \begin{bmatrix} x_k \\ v_k \end{bmatrix},\ \boldsymbol{A} = \begin{bmatrix} 1-b & a \\ -b & a \end{bmatrix},\ \boldsymbol{B} = \begin{bmatrix} b \\ b \end{bmatrix}.$$

Как известно, динамика линейной системы определяется собственными значениями матрицы динамики $\boldsymbol{A}$. В частности, необходимым и достаточным условием устойчивости положения равновесия $\boldsymbol{y}^* = [p,\, 0]^T$ будет $|\lambda_{1,2}| < 1$, где $\lambda_{1,2}$ – собственные числа матрицы $\boldsymbol{A}$. Легко показать, что условия $|\lambda_{1,2}| < 1$ равносильны

$$a < 1,\ b > 0,\ 2a - b + 2 > 0.$$

Эти неравенства определяют ограничения на весовые коэффициенты w_v, w_p, w_g, при которых динамика будет приводить к положению устойчивого равновесия.

В методе роя частиц векторы $\boldsymbol{p}_i$ и $\boldsymbol{g}$ иногда обновляются, с точки зрения теории выше это будет означать, что будет меняться положение равновесия линейной системы и последующая динамика. Впрочем, изменение положения равновесия никак не влияет на его тип устойчивости. Наконец, в методе роя частиц коэффициенты в когнитивной и социальной составляющей являются случайными, что приводит к стохастической динамике со случайной правой частью, но уравнения не перестают быть линейными.

Подробнее анализ решений полученных линейных уравнений в зависимости от параметров a, b можно найти в работе[22]. Теоретический анализ метода роя частиц можно найти в монографии[23].

5.3. Симплекс-метод Нелдера – Мида

Симплекс-метод Нелдера – Мида (*Nelder – Mead's method*) – это популярный безградиентный метод оптимизации, появившийся в 1965 году. Несмотря на схожее название, этот метод не следует путать с симплекс-методом Данцига (1947) для решения задачи линейного программирования. Метод Нелдера – Мида относится к методам решения задачи безусловной оптимизации:

$$f(\boldsymbol{x}) \to \min,\ \boldsymbol{x} \in \mathbb{R}^n.$$

Суть метода состоит в следующем. В n-мерном пространстве инициализируется многогранник (*симплекс*) с $(n+1)$-й вершиной. В зависимости от значений функции в его вершинах, на каждой итерации метода этот симплекс модифицируется в новый симплекс, перемещается в пространстве $\mathbb{R}^n$, его размеры могут меняться. На каждом шаге многогранник меняется так, чтобы он перемещался в сторону убывания значения функции.

Рассмотрим теперь один из алгоритмов этого метода[24]. Пусть на k-й итерации метода имеется симплекс с $(n+1)$-й вершиной. Предположим, что вершины $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n+1}$ упорядочены по возрастанию функции:

$$f(\boldsymbol{x}_1) \le \cdots \le f(\boldsymbol{x}_{n+1}).$$

[22] Trelea I.C. The particle swarm optimization algorithm: convergence analysis and parameter selection // Information processing letters. 2003. V. 85, N. 6. P. 317–325.

[23] Simon D. Evolutionary optimization algorithms. Wiley, 2013. P. 265.

[24] Gao F., Han L. Implementing the Nelder–Mead simplex algorithm with adaptive parameters // Computational Optimization and Applications. 2012. V. 51, N. 1. P. 259–277.

Вычислим среднее первых n точек симплекса

$$\boldsymbol{x}_{\text{ср}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$$

и найдем новую точку

$$\boldsymbol{x}_r = \boldsymbol{x}_{\text{ср}} + \rho(\boldsymbol{x}_{\text{ср}} - \boldsymbol{x}_{n+1}),$$

где $\rho > 0$, например, $\rho = 1$. Далее будем действовать в зависимости от ситуации.

1) Если $f(\boldsymbol{x}_r) < f(\boldsymbol{x}_1)$, то вычислить $\boldsymbol{x}_e = \boldsymbol{x}_{\text{ср}} + \chi(\boldsymbol{x}_{\text{ср}} - \boldsymbol{x}_{n+1})$ для некоторого $\chi > \rho$, например, $\chi = 2\rho$. Если $f(\boldsymbol{x}_e) < f(\boldsymbol{x}_r)$, то заменить $\boldsymbol{x}_{n+1}$ на $\boldsymbol{x}_e$ и назвать это действие *expand*. Если же $f(\boldsymbol{x}_e) \geq f(\boldsymbol{x}_r)$, то заменить $\boldsymbol{x}_{n+1}$ на $\boldsymbol{x}_r$ и назвать действие *reflect*.

2) Если $f(\boldsymbol{x}_1) \leq f(\boldsymbol{x}_r) < f(\boldsymbol{x}_n)$, то заменить $\boldsymbol{x}_{n+1}$ на $\boldsymbol{x}_r$ и назвать это действие *reflect*.

3) Если $f(\boldsymbol{x}_n) \leq f(\boldsymbol{x}_r) < f(\boldsymbol{x}_{n+1})$, то вычислить

$$\boldsymbol{x}_c = \boldsymbol{x}_{\text{ср}} + \gamma(\boldsymbol{x}_{\text{ср}} - \boldsymbol{x}_{n+1})$$

для некоторого положительного $\gamma < \rho$, например, $\gamma = \rho/2$. Если же выполнено $f(\boldsymbol{x}_c) \leq f(\boldsymbol{x}_r)$, то заменить $\boldsymbol{x}_{n+1}$ на $\boldsymbol{x}_c$ и назвать это действие *contract outside*. Иначе выполнить действие *shrink* (определение будет дано позднее).

4) Если $f(\boldsymbol{x}_{n+1}) \leq f(\boldsymbol{x}_r)$, то вычислить точку

$$\boldsymbol{x}_{cc} = \boldsymbol{x}_{\text{ср}} - \gamma(\boldsymbol{x}_{\text{ср}} - \boldsymbol{x}_{n+1}).$$

Если $f(\boldsymbol{x}_{cc}) < f(\boldsymbol{x}_{n+1})$, то заменить $\boldsymbol{x}_{n+1}$ на $\boldsymbol{x}_{cc}$, это действие называется *contract inside*. Иначе выполнить действие *shrink*.

Действие *shrink* состоит в расчете новых вершин:

$$\boldsymbol{v}_i = \boldsymbol{x}_1 + \sigma(\boldsymbol{x}_i - \boldsymbol{x}_1),\ i = 2, \ldots, n+1,$$

где $0 < \sigma < 1$ (например, $\sigma = 1/2$), а в качестве нового многогранника рассматривается многогранник с вершинами $\boldsymbol{x}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{n+1}$.

После этих операций точки снова упорядочиваются по значению функции и все повторяется заново. Алгоритм останавливается, когда достигается достаточно малое значение целевой функции, многогранник становится слишком малым или когда превышено число итераций.

Про метод Нелдера – Мида известно сравнительно немного результатов о сходимости или вообще о поведении многогранника. Вот

некоторые из фактов[25]. Предположим, что метод стартует с невырожденного многогранника (его объем ненулевой).

Теорема 1. *Пусть $f = f(x)$ – строго выпуклая на $\mathbb{R}$ функция. Тогда операция shrink не будет выполнена никогда.*

Теорема 2. *Пусть $f = f(x)$ – строго выпуклая на $\mathbb{R}$ функция, имеющая ограниченные множества уровней $\Gamma_\mu = \{x : f(x) \le \mu\}$. Пусть $\rho > 0$, $\chi > 1$, $\chi > \rho$, $\rho\chi \ge 1$, $0 < \gamma < 1$. Тогда обе точки интервала сходятся к решению оптимизационной задачи.*

Теорема 3. *Пусть $f = f(\boldsymbol{x})$ – строго выпуклая на $\mathbb{R}^2$ функция и имеет ограниченные множества уровней. Пусть $\rho = 1$, $\gamma = 1/2$. Тогда предельные значения функции в вершинах совпадают: $f_1^* = f_2^* = f_3^*$. Если положение лучшей с точки зрения функционала f точки не меняется от итерации к итерации, то есть $\boldsymbol{x}_1^{(k)} = \boldsymbol{x}_1^{(0)}$ постоянно, то симплекс сходится к этой точке $\boldsymbol{x}_1^{(0)}$.*

Теорема 4. *Пусть $f = f(\boldsymbol{x})$ – строго выпуклая на $\mathbb{R}^2$ функция и имеет ограниченные множества уровней. Пусть $\rho = 1$, $\gamma = 1/2$, $\chi = 2$. Тогда диаметр многогранника сходится к нулю с ростом числа итераций:*

$$\lim_{k\to\infty} \operatorname{diam} \Delta_k = 0.$$

Существуют примеры строго выпуклых функций на $\mathbb{R}^2$, для которых сходимость к оптимальной точке не гарантируется[26].

Практика показывает, что в задачах большой размерности метод Нелдера – Мида является неэффективным, существуют и теоретические обоснования этого наблюдения[27]. Отметим также, что хотя это безградиентный метод, это не значит, что его можно применять только для недифференцируемых функций. Целевая функция может быть всюду гладкая, но «шумная», с неровностями и «бугорками». Опыт показывает, что в этом случае градиентные методы могут не решить задачу, а метод Нелдера – Мида сойдется за малое число итераций.

[25]Lagarias J.C., Reeds J.A., Wright M.H. and Wright P.E. Convergence properties of the Nelder–Mead simplex method in low dimensions // SIAM Journal on optimization. 1998. V. 9, N. 1. P. 112–147.

[26]McKinnon K.I.M. Convergence of the Nelder-Mead simplex method to a nonstationary point // SIAM Journal on optimization. 1998. V. 9, N. 1. P. 148–158.

[27]Gao F., Han L. Implementing the Nelder–Mead simplex algorithm with adaptive parameters // Computational Optimization and Applications. 2012. V. 51, N. 1. P. 259–277.

Метод Нелдера – Мида реализован в MATLAB в функции `fminsearch`[28]. На Python он реализован в библиотеке `scipy` в функции `scipy.optimize.minimize` с опцией `method='Nelder-Mead'`[29].

5.4. Байесовская оптимизация

Метод байесовской оптимизации – это популярный безградиентный метод глобальной оптимизации вычислительно-затратных целевых функций. Это ценный для механики космического полета вид методов оптимизации, так как зачастую целевые функции или функции ограничения состоят из вычислительно-затратных процедур (например, интегрирования дифференциальных уравнений движения, итерационные процедуры) и могут быть зашемленными ошибками интегрирования или других численных методов. Байесовский метод оптимизации относится к *методам поверхностей отклика* (*surface response methods*), когда по известным точкам и значениям функции в этих точках строится аппроксимирующая целевую функцию поверхность, на которой тем или иным образом выбирается новая точка, от которой ожидается минимизация исходной целевой функции. В байесовской оптимизации аппроксимирующими поверхностями являются реализации гауссовских процессов.

Перейдем к описанию метода байесовской оптимизации. Решается оптимизационная задача с боксовыми ограничениями вида

$$f(\boldsymbol{x}) \to \min,\ x_i \in [a_i, b_i],\ i = 1, \ldots, n,$$

$$\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n.$$

Пусть имеются точки $\boldsymbol{x}_j$, $j = 1, \ldots, N$, удовлетворяющие ограничениям, и значения целевой функции в этих точках $f_j = f(\boldsymbol{x}_j)$. Одним из самых известных методов аппроксимации данных является метод линейной регрессии. В этом случае предполагается, что наблюдения удовлетворяют модели

$$y(\boldsymbol{x}_j) = \sum_k \beta_k f_k(\boldsymbol{x}_j) + \varepsilon_j.$$

Здесь функции $f_k(\boldsymbol{x}_j)$ могут быть линейными или нелинейными относительно $\boldsymbol{x}$, β_k – неизвестные коэффициенты, которые требуется

[28] https://www.mathworks.com/help/matlab/ref/fminsearch.html.

[29] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.mini\protect\discretionary{\char\hyphenchar\font}{}{}mize.html.

найти, а ε_k – нормально распределенные независимые в совокупности ошибки с нулевым средним и дисперсией σ^2. У такого подхода есть недостатки. Во-первых, необходимо заранее определить функции $f_k(\boldsymbol{x})$, то есть определить функциональную форму модели линейной регрессии. Во-вторых, эта модель может быть неадекватной для аппроксимации детерминированных непрерывных функций. А именно, в модели выше предполагается, что ошибки ε_j являются независимыми в совокупности, в то время как для близких точек $\boldsymbol{x}_k$ значения непрерывной функции тоже ожидаются близкими. Интуитивно понятно, что ошибки для близких $\boldsymbol{x}_k$ должны быть высоко коррелированы, а для далеких – менее коррелированы.

Чтобы учесть эти проблемы и особенности оптимизации непрерывных функций, предлагается другая модель:

$$y(\boldsymbol{x}_j) = \mu + \varepsilon(\boldsymbol{x}_j),$$

где μ – неизвестная постоянная, а $\varepsilon(\boldsymbol{x}_j)$ – случайные величины, зависящие от вектора $\boldsymbol{x}$, образующие нормально распределенный вектор $(\varepsilon(\boldsymbol{x}_1), ..., \varepsilon(\boldsymbol{x}_N))$ с нулевым средним и матрицей ковариаций:

$$C_{ij} = \operatorname{cov}(\varepsilon(\boldsymbol{x}_i), \varepsilon(\boldsymbol{x}_j)) = \sigma^2 \exp(-d(\boldsymbol{x}_i, \boldsymbol{x}_j)),$$

$$d(\boldsymbol{x}, \boldsymbol{z}) = \sum_{k=1}^{n} \theta_k |x_k - z_k|^{p_k},\ \theta_k \geq 0,\ p_k \in [1, 2].$$

Здесь функция d представляет собой расстояние между точками $\boldsymbol{x}$, $\boldsymbol{z}$, а коэффициенты θ_k, p_k неизвестны. Суммирование в функции d осуществляется по компонентам векторов $\boldsymbol{x}$ и $\boldsymbol{z}$.

В предлагаемой модели $2n + 2$ неизвестных, подлежащих поиску: μ, σ^2, θ_1, ..., θ_n, p_1, ..., p_n. Параметр μ отвечает за среднее значение оптимизируемой функции в области ограничений. Параметр σ^2 характеризует величину отклонений целевой функции от значения μ. Параметры θ_i отвечают за величину вклада i-й компоненты в рост расстояния. Например, если значение θ_i велико, то и расстояние d между векторами получится большим, а корреляционный момент между векторами малым. Параметры p_i отвечают за степень падения корреляционного момента при росте невязки между i-ми компонентами.

Отметим, что линейная регрессия сосредотачивается на регрессорах f_k и коэффициентах перед ними и делает простые предположения об ошибках модели. Предлагаемая же модель (она называется *DACE stochastic process model*[30] как акроним статьи Design and Analysis of

[30]Sacks J., Welch W.J., Mitchell T.J. and Wynn H.P. Design and analysis of

Computer Experiments, в которой этот подход был популяризирован) делает простые предположения о регрессорах (просто постоянная), но фокусируется на структуре корреляционных моментов ошибок. Можно даже сказать, что линейная регрессия отвечает на вопрос: «Чем является функция?», в то время как модель DACE отвечает на вопрос: «Каково типичное поведение функции?».

Вектор измерений $(y_1, \ldots, y_N)$ в рамках модели выше имеет нормальное распределение с плотностью вероятности

$$p(\boldsymbol{y}) = \frac{1}{(2\pi)^{N/2}\sqrt{\det \boldsymbol{C}}} \exp\left(-\frac{(\boldsymbol{y} - \mathbf{1}\mu)^T \boldsymbol{C}^{-1} (\boldsymbol{y} - \mathbf{1}\mu)}{2}\right),$$

где $\mathbf{1}$ – вектор из N единиц. Если зафиксировать коэффициенты θ_i и p_i, то эту плотность вероятности можно максимизировать[31] относительно μ и σ^2 и получить

$$\hat{\mu} = \frac{\mathbf{1}^T \boldsymbol{C}^{-1} \boldsymbol{y}}{\mathbf{1}^T \boldsymbol{C}^{-1} \mathbf{1}},\ \hat{\sigma}^2 = \frac{(\boldsymbol{y} - \mathbf{1}\hat{\mu})^T \boldsymbol{C}^{-1} (\boldsymbol{y} - \mathbf{1}\hat{\mu})}{n}.$$

Теперь, если поставить эти оценки обратно в функцию плотности, мы получим функцию параметров θ_i, p_i. Полученную функцию можно оптимизировать и найти оптимальные значения $\hat{\theta}_i$, $\hat{p}_i$ и соответствующую им матрицу ковариаций $\hat{\boldsymbol{C}}$. Затем, зафиксировав эти значения, можно снова найти новые оценки $\hat{\mu}$ и $\hat{\sigma}^2$ по формулам выше.

После того как неизвестные параметры найдены, можно оценить значение целевой функции в произвольной точке. Именно, из математической статистики известна оптимальная в средне-квадратичном смысле оценка

$$\hat{y}(\boldsymbol{x}) = \hat{\mu} + \frac{1}{\hat{\sigma}^2} \boldsymbol{r}(\boldsymbol{x})^T \hat{\boldsymbol{C}}^{-1} (\boldsymbol{y} - \mathbf{1}\hat{\mu}),$$

где вектор $\boldsymbol{r}(\boldsymbol{x})$ состоит из $r_i = \mathrm{cov}(\varepsilon(\boldsymbol{x}), \varepsilon(\boldsymbol{x}_i))$. Известно выражение и для дисперсии этой оценки[32]. Дисперсия равна нулю в точках $\boldsymbol{x}_j$.

---

computer experiments (with discussion) // Statistical Science. 1989. V. 4, N. 4. P. 409–435.

[31] Значения параметров, максимизирующие плотность вероятности измерений, называются в математической статистике *оценкой максимального правдоподобия*. Известны многочисленные свойства такой оценки неизвестных параметров распределения.

[32] Jones D. R., Schonlau M., Welch W. J. Efficient global optimization of expensive black-box functions // Journal of Global optimization. 1998. V. 13, N. 4. P. 462.

Перейдем теперь непосредственно к алгоритму оптимизации исходной функции. Пусть имеются данные $\{(\boldsymbol{x}_j, f_j)\}_{j=1}^N$ расчетов функции $f(\boldsymbol{x})$, $f_j = f(\boldsymbol{x}_j)$. Стоит задача поиска точки $\boldsymbol{x}_{N+1}$, такой, что $f(\boldsymbol{x}_{N+1}) \leq \min_{j=1,\ldots,N} f(\boldsymbol{x}_j)$. По данным $\{(\boldsymbol{x}_j, f_j)\}_{j=1}^N$ с помощью алгоритмов выше можно построить аппроксимацию $\hat{y}(\boldsymbol{x})$ функции $f(\boldsymbol{x})$. Так получится, что в некоторых областях значений $\boldsymbol{x}$ функция $f(\boldsymbol{x})$ будет известна с хорошей точностью (дисперсия $\hat{y}(\boldsymbol{x})$ мала в окрестности точек $\boldsymbol{x}_j$). В других же областях поведение функции $f(\boldsymbol{x})$ будет неопределенным (дисперсия $\hat{y}(\boldsymbol{x})$ высока далеко от точек $\boldsymbol{x}_j$). Возникает дилемма исследования и использования (exploration-exploitation dilemma): можно выбрать точку, наверняка минимизирующую функцию в области с высокой определенностью (использование), а можно выбрать ее в области с низкой определенностью и повысить шансы на поиск глобального оптимума (исследование). Дилемму исследования и использования можно решать различными способами. Как правило, она сводится к оптимизации некоторого вспомогательного функционала (*figure of merit*), например, *ожидаемого улучшения* (*expected improvement*):

$$I(\boldsymbol{x}) = \mathbb{E} \max(f_{\min} - \hat{y}(\boldsymbol{x}), 0).$$

На простых примерах можно увидеть, что эта функция имеет множество максимумов и равна нулю в точках измерений $\boldsymbol{x}_j$. Разработаны эффективные методы поиска максимума такой функции, они основываются на *алгоритмах ветвей и границ* (*branch-and-bound algorithms*), зарекомендовавших себя в комбинаторной оптимизации[33].

Итак, новая точка $\boldsymbol{x}_{N+1}$, от которой ожидается улучшение функционала, находится в результате решения некоторой подзадачи оптимизации. После того как она найдена, в ней рассчитывается значение исходного функционала $f_{N+1} = f(\boldsymbol{x}_{N+1})$, данные $\{(\boldsymbol{x}_j, f_j)\}$ пополняются новой точкой $(\boldsymbol{x}_{N+1}, f_{N+1})$, модель функции $\hat{y}(\boldsymbol{x})$ обновляется, процесс оптимизации продолжается поиском новой точки. Весь этот процесс и называется *байесовской оптимизацией*.

Перед запуском алгоритма байесовской оптимизации данные зачастую $\{(\boldsymbol{x}_j, f_j)\}$ генерируются методом Монте – Карло. Например, сначала можно сгенерировать 10–20 точек $\boldsymbol{x}_j$, посчитать в них значение функционала, а затем запустить алгоритм байесовской оптимизации.

[33] Clausen J. Branch and bound algorithms – principles and examples // Department of Computer Science, University of Copenhagen. 1999. P. 1–30.

Примеры применения байесовской оптимизации и более подробную информацию о методе можно найти в работе[34].

В MATLAB байесовская оптимизация реализована в функции `bayesopt`[35]. На Python известно несколько реализаций, среди них `BayesianOptimization`[36] и `BayesSearchCV`[37] пакета `skopt` библиотеки `scikit-optimize`. Реализация `BayesianOptimization` содержит полезный инструмент – *последовательное сокращение области поиска* (*sequential domain reduction*)[38]. Пользователь задает область поиска и коэффициент, с которым на каждой итерации будет происходить сокращение области к лучшему из найденных решений. Это помогает повысить точность искомого решения.

6. Методы решения дифференциальных уравнений

6.1. Введение

В механике космического полета орбитальное и вращательное движение аппарата описывается, как правило, системами обыкновенных дифференциальных уравнений (ОДУ). Перейдем к вопросу о том, какие методы обычно применяются для их решения.

В принципе методы решения обыкновенных дифференциальных уравнений можно разбить на три класса: точные, приближенные и численные[39].

К *точным* методам относятся методы, которые позволяют выразить решение дифференциального уравнения через элементарные функции, причем с конечным числом арифметических операций и композиций. Эти методы изучаются на втором курсе МФТИ в рамках годового курса по дифференциальным уравнениям и в данном пособии

[34] Jones D. R., Schonlau M., Welch W.J. Efficient global optimization of expensive black-box functions // Journal of Global optimization. 1998. V. 13, N. 4. P. 455–492.

[35] https://www.mathworks.com/help/stats/bayesopt.html.

[36] https://github.com/bayesian-optimization/BayesianOptimization.

[37] https://scikit-optimize.github.io/stable/modules/generated/skopt.BayesSearchCV.html.

[38] Stander N. and Craig K.J. On the robustness of a simple domain reduction scheme for simulation-based optimization // Engineering Computations. 2002. V. 19, N. 4. P. 431–450. URL: http://www.truegrid.com/srsm_revised.pdf.

[39] Калиткин Н.Н. Численные методы : учеб. пособие. 2-изд. Санкт-Петербург : БХВ-Петербург, 2014. С. 271.

рассматриваться не будут.

Приближенными называются методы, в которых решение исходной системы представляется как предел последовательности функций или как функциональный ряд. Причем для расчетов на компьютере этот ряд обрывают, поэтому используемое решение является «приближенным». Эти методы могут использоваться при качественном анализе сложной динамики аппарата и применяются в так называемой *теории возмущений*, элементы которой изучаются на других предметах на нашей базовой кафедре.

Наконец, *численные методы* – это алгоритмы вычисления искомого решения на некоторой выбранной сетке значений независимой переменной. Решение при этом получается в форме массива чисел, в том время как в точных и приближенных методах мы находим решения как функции, определенных на всем множестве значений независимой переменной. Численные методы не позволяют найти общего решения системы ОДУ, они могут дать только какое-то частное решение, например, решение задачи Коши. Можно считать это недостатком таких методов, но зато они применимы к очень широким классам уравнений и всем типам задач для них.

Прежде чем переходить к численным методам решения систем ОДУ, напомню постановку задачи Коши и некоторые сведения из теории дифференциальных уравнений о существовании и единственности ее решений.

6.2. Теоремы существования и единственности

Будем изучать системы ОДУ первого порядка вида

$$\begin{cases} \dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}), \\ \boldsymbol{y}(t_0) = \boldsymbol{y}_0 \end{cases} \tag{12}$$

на отрезке времени $t \in [t_0, t_f]$. Будем считать, что вектор-функции решения $\boldsymbol{y}(t) = [y_1(t), \ldots, y_n(t)]^T$ и правой части уравнения $\boldsymbol{f}(t, \boldsymbol{y}) = [f_1(t, \boldsymbol{y}), \ldots, f_n(t, \boldsymbol{y})]^T$ имеют в общем случае размерность $n \geq 1$.

Вспомним сначала условия, при которых решение системы (12) существует и единственно. Для этого введем на отрезке интегрирования $t \in [t_0, t_f]$ сетку

$$t_0 = t_1 < t_2 \cdots < t_N = t_f$$

из $N \geq 2$ узлов. В пространстве $\mathbb{R}^n$ введем множество точек

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + (t_{k+1} - t_k)\boldsymbol{f}(t_k, \boldsymbol{y}_k), \ k = 1, \ldots, N-1.$$

Кусочно-линейную функцию

$$\boldsymbol{y}_h(t) = \boldsymbol{y}_k + (t - t_k)\boldsymbol{f}(t_k, \boldsymbol{y}_k),\ t_k \le t \le t_{k+1},\ k = 1, \ldots, N-1$$

будем называть *ломаной Эйлера*. Введем также обозначение

$$|h| = \max_{k=1,\ldots,N-1}(t_{k+1} - t_k).$$

Будем называть эту величину *мелкостью сетки*.

Всюду далее *приближенные* значения решения $\boldsymbol{y}(t)$ на сетке мы будем обозначать как

$$\boldsymbol{y}_k = [y_{1,k}, \ldots, y_{n,k}]^T,\ k = 1, \ldots, N,$$

где первый индекс нумерует компоненты вектора $\boldsymbol{y}_k$, а второй индекс указывает на номер узла сетки $k = 1, \ldots, N$. Точные же решения решения будем обозначать просто как $\boldsymbol{y}(t_k)$.

Теорема 1[40]. *Пусть на множестве*

$$D = \{(t, \boldsymbol{y}) :\ t_0 \le t \le t_f,\ |\boldsymbol{y} - \boldsymbol{y}_0| \le R\}$$

функция $\boldsymbol{f}(t, \boldsymbol{y})$ непрерывна, ограничена по модулю сверху числом M и удовлетворяет условию Липшица:

$$\exists L > 0 :\ \forall t \in [t_0, t_f],\ \forall \boldsymbol{y}, \boldsymbol{z} \in D\ \ |\boldsymbol{f}(t, \boldsymbol{z}) - \boldsymbol{f}(t, \boldsymbol{y})| \le L|\boldsymbol{z} - \boldsymbol{y}|. \quad (13)$$

Тогда, если $t_f - t_0 \le R/M$, то 1) при $|h| \to 0$ ломаные Эйлера сходятся равномерно к некоторой непрерывной функции $\boldsymbol{\varphi}(t)$, 2) функция $\boldsymbol{\varphi}(t)$ непрерывно дифференцируема и является решением задачи (12) *на отрезке $[t_0, t_f]$, 3) при $t_0 \le t \le t_f$ задача* (12) *не имеет других решений.*

Теорема 2[41]. *Пусть U является открытым множеством в $\mathbb{R}^{n+1}$, а $\boldsymbol{f}$ и $\partial\boldsymbol{f}/\partial\boldsymbol{y}$ непрерывны в U. Тогда для любой точки $(t_0, \boldsymbol{y}_0) \in U$ существует единственное решение задачи* (12)*, которое можно продолжить до границы U (в обоих направлениях).*

Постоянная L в теоремах выше называется *константой Липшица*. Заметим, что из ограниченности производной функции $\boldsymbol{f}$ следует свойство «липшицевости» (13), причем в качестве константы Липшица

[40] Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи. Москва : Мир, 1990. С. 39, 57 (теорема 7.3).

[41] Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи. Москва : Мир, 1990. С. 42, 57 (теорема 7.4).

можно взять норму матрицы $\partial \boldsymbol{f}/\partial \boldsymbol{y}$, подчиненную норме $\boldsymbol{y}$. Если норму матрицы вычислить трудно, но известно, что $|\partial f_i/\partial y_j| \le q_{ij}$, то для случая квадратичной нормы $\|\cdot\|_2$ в качестве L можно взять[42]

$$L = \sqrt{\sum_{i,j} q_{ij}^2}.$$

Для нормы $\|\cdot\|_\infty$ можно принять $L = \max\limits_i \sum_j |q_{ij}|$, а для нормы $\|\cdot\|_1$ можно взять $L = \max\limits_j \sum_i |q_{ij}|$.

6.3. Константа Липшица в задаче двух тел

Рассмотрим уравнения движения в модели задачи двух тел:

$$\dot{\boldsymbol{r}} = \boldsymbol{v},\ \dot{\boldsymbol{v}} = -\mu \boldsymbol{r}/r^3.$$

Найдем константу Липшица правой части уравнений в области

$$|\boldsymbol{r}| \ge r_{\min}.$$

Запишем эти уравнения в виде $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$, где

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{v} \end{bmatrix},\ \boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{v} \\ -\mu \boldsymbol{r}/r^3 \end{bmatrix}.$$

Производная правых частей уравнения по фазовому вектору равна

$$\boldsymbol{J} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \boldsymbol{O}_{3\times 3} & \boldsymbol{E}_3 \\ \mu\left(\dfrac{3\boldsymbol{r}\boldsymbol{r}^T}{r^5} - \dfrac{\boldsymbol{E}_3}{r^3}\right) & \boldsymbol{O}_{3\times 3} \end{bmatrix},$$

как было найдено ранее. В качестве константы Липшица найдем норму этой матрицы, подчиненную норме $\|\boldsymbol{x}\|_2 = \sqrt{x_1^2 + \cdots + x_6^2}$, она выражается формулой

$$L = \sqrt{\lambda_{\max}(\boldsymbol{J}^T\boldsymbol{J})},$$

где под знаком корня находится максимальное собственное число матрицы $\boldsymbol{J}^T\boldsymbol{J}$. Запишем эту матрицу:

$$\boldsymbol{J}^T\boldsymbol{J} = \begin{bmatrix} \boldsymbol{E}_3 & \boldsymbol{O}_{3\times 3} \\ \boldsymbol{O}_{3\times 3} & \mu^2\left(\dfrac{3\boldsymbol{r}\boldsymbol{r}^T}{r^5} - \dfrac{\boldsymbol{E}_3}{r^3}\right)^T\left(\dfrac{3\boldsymbol{r}\boldsymbol{r}^T}{r^5} - \dfrac{\boldsymbol{E}_3}{r^3}\right) \end{bmatrix}.$$

[42] Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи. Москва : Мир, 1990. С. 57.

Ее максимальное собственное значение равно

$$\lambda_{\max}(\boldsymbol{J}^T\boldsymbol{J}) = \max\left(1, \lambda_{\max}\left(\mu^2\left(\frac{3\boldsymbol{r}\boldsymbol{r}^T}{r^5} - \frac{\boldsymbol{E}_3}{r^3}\right)^T\left(\frac{3\boldsymbol{r}\boldsymbol{r}^T}{r^5} - \frac{\boldsymbol{E}_3}{r^3}\right)\right)\right) =$$

$$= \max\left(1, |\lambda|^2_{\max}\left(\mu\left(\frac{3\boldsymbol{r}\boldsymbol{r}^T}{r^5} - \frac{\boldsymbol{E}_3}{r^3}\right)\right)\right),$$

так как матрица $3\boldsymbol{r}\boldsymbol{r}^T/r^5 - \boldsymbol{E}_3/r^3$ симметричная. Вторым аргументом функции max является квадрат максимального по модулю собственного значения матрицы. Немного упростим выражение:

$$\lambda_{\max}(\boldsymbol{J}^T\boldsymbol{J}) = \max\left(1, \mu^2|\lambda|^2_{\max}\left(\frac{3\boldsymbol{r}\boldsymbol{r}^T}{r^5} - \frac{\boldsymbol{E}_3}{r^3}\right)\right).$$

Теперь изучим матрицу $3\boldsymbol{r}\boldsymbol{r}^T/r^5 - \boldsymbol{E}_3/r^3$:

$$\frac{3\boldsymbol{r}\boldsymbol{r}^T}{r^5} - \frac{\boldsymbol{E}_3}{r^3} = \begin{bmatrix} 3x^2/r^5 - 1/r^3 & 3xy/r^5 & 3xz/r^5 \\ 3xy/r^5 & 3y^2/r^5 - 1/r^3 & 3yz/r^5 \\ 3xz/r^5 & 3yz/r^5 & 3z^2/r^5 - 1/r^3 \end{bmatrix}.$$

Так как движение по орбите в модели задачи двух тел всегда остается в одной плоскости, перейдем с помощью ортогонального преобразования в систему координат, в которой всегда будет выполнено $z = 0$. В результате ортогонального преобразования собственные числа матрицы не изменятся. Получаем

$$\frac{3\boldsymbol{r}\boldsymbol{r}^T}{r^5} - \frac{\boldsymbol{E}_3}{r^3} = \begin{bmatrix} 3x^2/r^5 - 1/r^3 & 3xy/r^5 & 0 \\ 3xy/r^5 & 3y^2/r^5 - 1/r^3 & 0 \\ 0 & 0 & -1/r^3 \end{bmatrix}.$$

Одно из собственных чисел этой матрицы равно $\lambda_1 = -1/r^3$. Два других собственных числа находятся из квадратного уравнения

$$\lambda^2 - 1/r^3 \cdot \lambda + (3x^2/r^5 - 1/r^3)(3y^2/r^5 - 1/r^3) - 9x^2y^2/r^{10} = 0.$$

Здесь мы воспользовались равенством $x^2 + y^2 = r^2$, учитывая, что $z = 0$. Упрощая уравнение выше, получаем

$$\lambda^2 - 1/r^3 \cdot \lambda - 2/r^6 = 0.$$

Корнями этого уравнения являются $\lambda_2 = 2/r^3$ и $\lambda_3 = -1/r^3$. Максимальное по модулю собственное значение равно $2/r^3$, а во всей области $\|\boldsymbol{r}\| \geq r_{\min}$ оно равно $2/r^3_{\min}$. В результате мы получаем, что

$$L = \sqrt{\lambda_{\max}(\boldsymbol{J}^T\boldsymbol{J})} = \max\left(1, \frac{2\mu}{r^3_{\min}}\right).$$

6.4. Методы Рунге – Кутты

Общая схема *методов Рунге – Кутты* выражается формулой

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + h_k(b_1\boldsymbol{k}_1 + \cdots + b_s\boldsymbol{k}_s),$$

где вспомогательные векторы

$$\begin{aligned}
&\boldsymbol{k}_1 = \boldsymbol{f}(t_k + hc_1, \boldsymbol{y}_k + h_k(a_{11}\boldsymbol{k}_1 + \cdots + a_{1s}\boldsymbol{k}_s)),\\
&\boldsymbol{k}_2 = \boldsymbol{f}(t_k + hc_2, \boldsymbol{y}_k + h_k(a_{21}\boldsymbol{k}_1 + \cdots + a_{2s}\boldsymbol{k}_s)),\\
&\dots\\
&\boldsymbol{k}_s = \boldsymbol{f}(t_k + hc_s, \boldsymbol{y}_k + h_k(a_{s1}\boldsymbol{k}_1 + \cdots + a_{ss}\boldsymbol{k}_s))
\end{aligned}$$

называются *стадиями метода*. Здесь $h_k = t_{k+1} - t_k$ – длина шага интегрирования, а вещественные числа a_{ij}, b_i, c_i подбираются так, чтобы удовлетворить некоторым требованиям (например, на порядок аппроксимации). Расчеты начинаются при $k = 1$ с использованием начального условия $\boldsymbol{y}_0 = \boldsymbol{y}_1$, в результате чего получается вектор $\boldsymbol{y}_2$. Затем при $k = 2$ с использованием $\boldsymbol{y}_2$ вычисляется $\boldsymbol{y}_3$ и так далее. Заметим, что расчет стадий $\boldsymbol{k}_1, \dots, \boldsymbol{k}_s$ представляет собой решение системы ns алгебраических уравнений, где n – размерность вектора $\boldsymbol{y}$. Решается такая система обычно численно: методами простой итерации, методом Ньютона или как-то иначе.

Если $a_{ij} = 0$ для всех $j \geq i$, то соответствующий метод называется *явным*, иначе – *неявным*. Если метод явный, то стадии рассчитываются последовательно: сначала вычисляется $\boldsymbol{k}_1$, затем на основе $\boldsymbol{k}_1$ вычисляется $\boldsymbol{k}_2$, затем на основе $\boldsymbol{k}_1$ и $\boldsymbol{k}_2$ вычисляется $\boldsymbol{k}_3$, и так далее.

Любому методу Рунге – Кутты можно поставить в соответствие таблицу

c_1	a_{11}	a_{12}	$\dots$	a_{1s}
c_2	a_{21}	a_{22}	$\dots$	a_{2s}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
c_s	a_{s1}	a_{s2}	$\dots$	a_{ss}
	b_1	b_2	$\dots$	b_s

Она называется *таблицей Бутчера*, а матрицу $\{a_{ij}\}$ называют *матрицей Бутчера*. Приведем схему самого простого метода Рунге – Кутты:

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + h\boldsymbol{f}(t_k, \boldsymbol{y}_k),$$

который еще называют *явным методом Эйлера*. В наших обозначениях его можно записать в виде

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + h\boldsymbol{k}_1,\ \boldsymbol{k}_1 = \boldsymbol{f}(t_k, \boldsymbol{y}_k),$$

то есть $s = 1$, $c_1 = 0$, $b_1 = 1$, $a_{11} = 0$. Таблица Бутчера для него: $\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$

Еще есть *неявный метод Эйлера*:

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + h\boldsymbol{f}(t_{k+1}, \boldsymbol{y}_{k+1}),$$

который можно записать в виде

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + h\boldsymbol{k}_1,\ \boldsymbol{k}_1 = \boldsymbol{f}(t_k + h, \boldsymbol{y}_k + h\boldsymbol{k}_1),$$

то есть $s = 1$, $c_1 = 1$, $b_1 = 1$, $a_{11} = 1$. Таблица Бутчера для него: $\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$

Приведем также схему одного из самых популярных методов Рунге – Кутты – *классического четырехстадийного явного метода Рунге – Кутты*:

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \frac{h}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4)$$

со стадиями

$$\begin{aligned}
\boldsymbol{k}_1 &= \boldsymbol{f}(t_k, \boldsymbol{y}_k), \\
\boldsymbol{k}_2 &= \boldsymbol{f}(t_k + h/2, \boldsymbol{y}_k + h\boldsymbol{k}_1/2), \\
\boldsymbol{k}_3 &= \boldsymbol{f}(t_k + h/2, \boldsymbol{y}_k + h\boldsymbol{k}_2/2), \\
\boldsymbol{k}_4 &= \boldsymbol{f}(t_k + h, \boldsymbol{y}_k + h\boldsymbol{k}_3).
\end{aligned}$$

Этому методу соответствует таблица Бутчера:

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

Говорят, что *метод Рунге – Кутты имеет порядок* p, если для всех достаточно гладких задач (12)

$$\|\boldsymbol{y}(t_2) - \boldsymbol{y}_2\| = O(h^{p+1}),\ h \to 0,$$

где $\boldsymbol{y}(t_2)$ – точное решение в точке t_2, а $\boldsymbol{y}_2$ – приближенное решение в той же точке, подсчитанное методом для точного начального условия $\boldsymbol{y}_1 = \boldsymbol{y}(t_1)$. Другими словами, метод имеет порядок p, если ряды Тейлора для $\boldsymbol{y}(t_2)$ и $\boldsymbol{y}_2$ совпадают до члена h^p включительно. Величину $\boldsymbol{e} = \boldsymbol{y}(t_2) - \boldsymbol{y}_2$ называют *локальной погрешностью метода*. В таблице 1 приведены несколько известных методов Рунге – Кутты и их порядки.

Теорема 3[43]. *Если метод Рунге – Кутты имеет порядок p и все частные производные $\boldsymbol{f}(t, \boldsymbol{y})$ непрерывно дифференцируемы $p+1$ раз, то*

$$\boldsymbol{y}(t_2) - \boldsymbol{y}_2 = \boldsymbol{C}h^{p+1} + O(h^{p+2}),\ h \to 0,$$

где вектор $\boldsymbol{C}$ не зависит от h, но зависит от коэффициентов метода и частных производных функции $\boldsymbol{f}(t, \boldsymbol{y})$ в точке $(t_1, \boldsymbol{y}_1)$.

Про связь порядка метода с количеством стадий известно[44], что если явный s-стадийный метод имеет порядок p, то $s \geq p$, а если $p \geq 5$, то $s > p$. Кроме того, известна[45] следующая связь между порядком метода и минимальным количеством стадий, достаточным для существования метода с порядком p:

p	1	2	3	4	5	6	7	8
$\min s$	1	2	3	4	6	7	9	11

Известно, что для всех $s \geq 1$ существуют неявные методы Рунге – Кутты порядка $p = 2s$. К таким методам относятся методы Гаусса – Лежандра[46], к которым, в частности, относятся неявный метод средней точки и следующий метод 4 порядка:

$1/2 - \sqrt{3}/6$	$1/4$	$1/4 - \sqrt{3}/6$
$1/2 + \sqrt{3}/6$	$1/4 + \sqrt{3}/6$	$1/4$
	$1/2$	$1/2$

[43] Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи. Москва : Мир, 1990. С. 168 (теорема 3.4).

[44] Butcher J.C. Numerical Methods for Ordinary Differential Equations. John Wiley & Sons, 2008. P. 187.

[45] Butcher J.C. Numerical Methods for Ordinary Differential Equations. John Wiley & Sons, 2008. P. 187–196.

[46] Iserles A.A First Course in the Numerical Analysis of Differential Equations. Cambridge University Press, 1996. P. 47.

Таблица 1. Некоторые известные методы Рунге – Кутты и их порядки

Метод	Таблица Бутчера
Явный метод Эйлера (порядок 1)	$\begin{array}{c\|c} 0 & 0 \\ \hline & 1 \end{array}$
Неявный метод Эйлера (порядок 1)	$\begin{array}{c\|c} 1 & 1 \\ \hline & 1 \end{array}$
Явный метод средней точки (порядок 2)	$\begin{array}{c\|cc} 0 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ \hline & 0 & 1 \end{array}$
Неявный метод средней точки (порядок 2)	$\begin{array}{c\|c} 1/2 & 1/2 \\ \hline & 1 \end{array}$
Классический метод Рунге–Кутты (порядок 4)	$\begin{array}{c\|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$
Правило 3/8 (порядок 4)	$\begin{array}{c\|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 0 \\ 2/3 & -1/3 & 1 & 0 & 0 \\ 1 & 1 & -1 & 1 & 0 \\ \hline & 1/8 & 3/8 & 3/8 & 1/8 \end{array}$

6.5. Сходимость методов Рунге – Кутты

Введем понятие *глобальной погрешности*:

$$\boldsymbol{e}_g = \boldsymbol{y}(t_f) - \boldsymbol{y}_N.$$

Теорема 4[47]. *Обозначим за U окрестность кривой*

$$\{(t, \boldsymbol{y}(t)) : \ t_0 \le t \le t_f\},$$

где $\boldsymbol{y}(t)$ – точное решение (12). Пусть в U справедливы оценки локальных погрешностей $\|\boldsymbol{e}_k\| = O(h^{p+1})$ и выполнено условие

$$\left\| \begin{array}{cc} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{f}}{\partial t} \\ 0 & 1 \end{array} \right\| \le L.$$

Тогда для достаточного малого h имеет место следующая оценка глобальной погрешности:

$$\|\boldsymbol{e}_g\| \le |h|^p \cdot \frac{C}{L} \left(e^{L(t_f - t_0)} - 1 \right),$$

где $|h|$ – мелкость сетки, C не зависит от h, то есть $\|\boldsymbol{e}_g\| = O(h^p)$, $h \to 0$.

6.6. Вложенные методы Рунге – Кутты

Рассмотрим теперь вопрос управления шагом интегрирования для обеспечения точности получаемого решения. Формула для локальной погрешности $\boldsymbol{e} = \boldsymbol{y}(t_2) - \boldsymbol{y}_2$ содержит точное значение решения $\boldsymbol{y}(t_2)$, но оно нам не известно. Поэтому вектор $\boldsymbol{e}$ мы вычислить не можем, хотя можем попробовать его оценить. Из теорем из предыщущих разделов мы знаем, что локальная погрешность равна

$$\boldsymbol{e} = \boldsymbol{C} h^{p+1} + O(h^{p+2}), \ h \to \infty,$$

где h – длина шага интегрирования, а постоянная $\boldsymbol{C}$ не зависит от h и выражается через производные функции $\boldsymbol{f}(t, \boldsymbol{y})$. Отсюда следует, что шаг можно было бы выбирать таким образом, чтобы $\|\boldsymbol{e}\| = \|\boldsymbol{C}\| h^{p+1} < < \varepsilon$, оценив сверху $\|\boldsymbol{C}\|$, но это не практично, так как выражение для

[47] Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи. Москва : Мир, 1990. С. 171 (теорема 3.4).

$\boldsymbol{C}$ трудно считать из-за наличия в нем большого числа производных высокой степени.

Существует несколько практических способов оценки локальной погрешности и управления шагом интегрирования. Мы рассмотрим способ, основанный на *вложенных* методах Рунге – Кутты: для оценки локальной погрешности вместо точного решения будем использовать приближенное решение, полученное другим методом Рунге – Кутты более высокого (или более низкого) порядка. Для экономии вычислений этот второй метод выбирается так, чтобы его a- и c-коэффициенты совпадали с a- и c-коэффициентами основного метода. Тогда стадии ($\boldsymbol{k}$-векторы) основного и этого вспомогательного метода совпадают.

Итак, пусть рассматриваются две схемы

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + h(b_1\boldsymbol{k}_1 + \cdots + b_s\boldsymbol{k}_s),$$

$$\hat{\boldsymbol{y}}_{k+1} = \boldsymbol{y}_k + h(\hat{b}_1\boldsymbol{k}_1 + \cdots + \hat{b}_s\boldsymbol{k}_s)$$

с совпадающими стадиями. Такие методы называются *вложенными* методами Рунге – Кутты, впервые они были найдены Р. Мерсоном в 1957 году[48] (см. Таблицу 2).

Если основной метод имеет порядок p, а вспомогательный – порядок $\hat{p}$, то соответствующий вложенный метод называют вложенным методом Рунге – Кутты порядка $p(\hat{p})$. При использовании вложенных методов локальная погрешность на k-м шаге оценивается по формуле

$$\boldsymbol{e} \approx \boldsymbol{y}_{k+1} - \hat{\boldsymbol{y}}_{k+1} = h\boldsymbol{k}_1(b_1 - \hat{b}_1) + \cdots + h\boldsymbol{k}_s(b_s - \hat{b}_s).$$

Таблица Бутчера метода Мерсона приведена ниже. Первая строчка b-коэффициентов отвечает основному методу, а вторая – вспомогательному. Основной метод имеет порядок 4, а вспомогательный – порядок 5 *для линейных уравнений с постоянными коэффициентами.*

Разберем теперь метод контроля точности решения. Для того чтобы контролировать точность, сначала для каждой компоненты $j = 1, \ldots, n$ вектора $\boldsymbol{y}$ задают допустимые значения абсолютной ε_j^a и относительной ε_j^r погрешности. Мы хотим, чтобы на каждом шаге интегрирования были выполнены неравенства

$$|y_{j,k+1} - \hat{y}_{j,k+1}| \le \underbrace{\varepsilon_j^a + \max(|y_{j,k}|, |y_{j,k+1}|)\varepsilon_j^r}_{\varepsilon_j}, \quad j = 1, \ldots, n,$$

[48] Merson R.H. An operational method for the study of integration processes // Proc. Symp. Data Processing, Weapons Research Establishment, Salisbury, Australia. P. 110-1 to 110-25.

Таблица 2. Метод Мерсона 4(«5»)

0	0	0	0	0	0
1/3	1/3	0	0	0	0
1/3	1/6	1/6	0	0	0
1/2	1/8	0	3/8	0	0
1	1/2	0	−3/2	2	0
	1/2	0	−3/2	2	0
	1/6	0	0	2/3	1/6

что равносильно неравенствам

$$E_j = \frac{|y_{j,k+1} - \hat{y}_{j,k+1}|}{\varepsilon_j} \le 1, \quad j = 1, \ldots, n.$$

Если все эти неравенства выполнены, то следует принять шаг и двигаться дальше. Если же какое-то неравенство не выполнено, следует отклонить шаг и уменьшить его длину на такую величину, чтобы все неравенства были выполнены. Посмотрим, как это можно сделать.

Сформируем вектор $\boldsymbol{E} = [E_1, \ldots, E_n]$ и будем рассматривать его норму $\|\boldsymbol{E}\|_\infty$. Так как $y_{j,k+1} \approx y_j(t_{k+1}) + C_j h^{p+1}$ и $\hat{y}_{j,k+1} \approx y_j(t_{k+1}) + \hat{C}_j h^{\hat{p}+1}$, то

$$\|\boldsymbol{E}\|_\infty \approx D \cdot h^{\min(p,\,\hat{p})+1}, \quad h \to 0.$$

Нам же нужен такой шаг h_{new}, чтобы $\|\boldsymbol{E}\|_\infty = 1$. Отсюда получаем

$$h_{new} = \left(\frac{1}{\|\boldsymbol{E}\|_\infty}\right)^{\frac{1}{\min(p,\hat{p})+1}}. \tag{14}$$

Этой формулой можно пользоваться, чтобы увеличивать длину шага, если она оказывается слишком маленькой ($\|\boldsymbol{E}\|_\infty \ll 1$), и уменьшать длину шага, если она оказывается большой ($\|\boldsymbol{E}\|_\infty > 1$). На практике в формулу (14) добавляют смягчающий множитель α, чтобы избежать вероятного отклонения шага в следующей итерации:

$$h_{new} = \alpha \cdot \left(\frac{1}{\|\boldsymbol{E}\|_\infty}\right)^{\frac{1}{\min(p,\hat{p})+1}}.$$

Обычно $\alpha = 0.8, 0.9, 0.95$.

Сегодня известно множество вложенных методов Рунге – Кутты. Одним из самых известных является метод Дорма́на – Принса 5(4)[49],

[49] Dormand J.R., Prince P.J. A family of embedded Runge-Kutta formulae // J. Comp. Appl. Math. 1980. V. 6. P. 19–26.

реализованный, например, в функции `ode45` на MATLAB и как класс `scipy.integrate.RK45` пакета `scipy` на языке Python. Метод пятого порядка используется как основной метод, именно его решение мы получаем на выходе, а метод четвертого порядка используется как вспомогательный для оценки локальной погрешности. Примечательно, что до 1990 г. в MATLAB функция `ode45` реализовывала метод Фельберга 4(5), разработанный в NASA в 1969 г.[50]. Метод Дормана – Принса отличается от метода Фельберга не только тем, что в качестве основного он берет метод более высокого порядка, но и тем, что он минимизирует *коэффициенты погрешностей* (грубо говоря, константу в $O(h^{p+1})$) для решения высокого порядка, а Фельберг минимизировал коэффициенты ошибок для решения низкого порядка. Численные эксперименты убедительно показали, что метод Дормана – Принса гораздо эффективнее метода Фельберга, поэтому от последнего и отказались[51].

Замечу, что метод Дормана – Принса имеет семь стадий, хотя начиная со второго шага обращений к функции правых частей $\boldsymbol{f}$ всего шесть. Так получается из-за того, что метод Дормана – Принса обладает свойством FSAL (First Same As Last): вектор $\boldsymbol{k}_7$ на текущем шаге совпадает с вектором $\boldsymbol{k}_1$ на следующем шаге, поэтому его рассчитывать не приходится.

6.7. Многошаговые методы Адамса

В методах Рунге – Кутты оценка решения $\boldsymbol{y}_{k+1}$ производится с использованием решения $\boldsymbol{y}_k$ на предыдущем узле. Существуют методы, которые для оценки $\boldsymbol{y}_{k+1}$ используют решения на нескольких предыдущих узлах, то есть $\boldsymbol{y}_k$, $\boldsymbol{y}_{k-1}$, $\boldsymbol{y}_{k-2}$ и так далее. Это так называемые *многошаговые* методы. Методы Рунге – Кутты поэтому можно назвать *одношаговыми* методами.

Мы рассмотрим только один класс многошаговых методов – методы Адамса, названные в честь английского астронома Джона Адамса. Эти методы были известны еще в 1855 году, а методы Рунге – Кутты появились около 1900 года. Кроме методов Адамса, к многошаговым методам относятся, например, так называемые *формулы дифференцирования назад* (ФДН) и вообще *общие линейные методы*[52].

[50] Fehlberg E. Low-order classical Runge–Kutta formulas with stepsize control and their application to some heat transfer problems. NASA Technical Report 315. 1969.

[51] Shampine L.F., Reichelt M.W. The MATLAB ODE Suite. 1997.

[52] Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи. Москва : Мир, 1990.

Пусть имеется равномерная сетка

$$t_0 = t_1 < t_2 < \cdots < t_N = t_f$$

на отрезке $[t_0, t_f]$. Обозначим $h = t_{k+1} - t_k$, $k = 1, \ldots, N-1$. Общая схема методов Адамса с постоянным шагом имеет вид

$$\begin{aligned} \boldsymbol{y}_{k+1} = \boldsymbol{y}_k + h\sum_{i=0}^{s}\beta_i(s)\boldsymbol{f}_{k+1-i} = \\ = \boldsymbol{y}_k + h\sum_{i=1}^{s}\beta_i(s)\boldsymbol{f}_{k+1-i} + \beta_0(s)\boldsymbol{f}_{k+1}, k \geq s, \end{aligned} \tag{15}$$

где $\beta_i = \beta_i(s)$ – коэффициенты, зависящие от s, а $\boldsymbol{f}_i = \boldsymbol{f}(t_i, \boldsymbol{y}_i)$. В этом выражении сумма состоит в общем случае из $s+1$ слагаемого, которые рассчитываются по значениям $\boldsymbol{y}_{k+1}, \boldsymbol{y}_k, \ldots, \boldsymbol{y}_{k+1-s}$. Если $\beta_0 = 0$, то метод называется явным, потому что расчет $\boldsymbol{y}_{k+1}$ можно произвести по этой формуле напрямую. В литературе этот класс методов называют также *методами Адамса – Башфорта*. Если же $\beta_0 \neq 0$, то $\boldsymbol{y}_{k+1}$ входит и в левую, и в правую часть этого уравнения. Такие методы называются неявными или *методами Адамса – Мултона*.

Напомню, что в методах Рунге – Кутты коэффициенты методов ищутся из условия на порядок метода. В методах Адамса коэффициенты ищутся из других соображений: так, чтобы $\boldsymbol{y}_{k+1}$ было точным решением системы дифференциальных уравнений

$$\dot{\boldsymbol{y}} = \boldsymbol{p}(t),\ \boldsymbol{y}(t_0) = \boldsymbol{y}_0,$$

где $\boldsymbol{p}(t)$ – полином переменной t, проходящий через точки $(t_{k+1}, \boldsymbol{f}_{k+1})$, $(t_k, \boldsymbol{f}_k)$ ..., $(t_{k+1-s}, \boldsymbol{f}_{k+1-s})$.

Определим понятие локальной погрешности метода Адамса:

$$\boldsymbol{e} = \boldsymbol{y}(t_{k+1}) - \boldsymbol{y}_{k+1},$$

где $\boldsymbol{y}(t_{k+1})$ – точное решение задачи Коши в точке t_{k+1}, а $\boldsymbol{y}_{k+1}$ – численное решение, рассчитанное методом для *точных* стартовых значений $\boldsymbol{y}_k = \boldsymbol{y}(t_k)$, ..., $\boldsymbol{y}_{k+1-s} = \boldsymbol{y}(t_{k+1-s})$, то есть стартовых значений, находящихся на точном решении задачи в точках $t_k, \ldots, t_{k+1-s}$ соответственно. Говорят, что метод Адамса имеет порядок p, если для любых достаточно гладких функций $\boldsymbol{f}$ будет $\|\boldsymbol{e}\| = O(h^{p+1})$ при $h \to 0$.

Приведем примеры явных и неявных методов Адамса для нескольких первых значений s. Для удобства во всех формулах вместо обобщенных точек $k+1$ мы рассматриваем первые приближенные значения искомого решения.

Явные методы Адамса:

$$s=1: \quad \boldsymbol{y}_2 = \boldsymbol{y}_1 + h\boldsymbol{f}_1, \qquad \text{(1-й порядок)}$$

$$s=2: \quad \boldsymbol{y}_3 = \boldsymbol{y}_2 + h(\tfrac{3}{2}\boldsymbol{f}_2 - \tfrac{1}{2}\boldsymbol{f}_1), \qquad \text{(2-й порядок)}$$

$$s=3: \quad \boldsymbol{y}_4 = \boldsymbol{y}_3 + h(\tfrac{23}{12}\boldsymbol{f}_3 - \tfrac{16}{12}\boldsymbol{f}_2 + \tfrac{5}{12}\boldsymbol{f}_1), \qquad \text{(3-й порядок)}$$

$$s=4: \quad \boldsymbol{y}_5 = \boldsymbol{y}_4 + h(\tfrac{55}{24}\boldsymbol{f}_4 - \tfrac{59}{24}\boldsymbol{f}_3 + \tfrac{37}{24}\boldsymbol{f}_2 - \tfrac{9}{24}\boldsymbol{f}_1). \qquad \text{(4-й порядок)}$$

Неявные методы Адамса:

$$s=0: \quad \boldsymbol{y}_2 = \boldsymbol{y}_1 + h\boldsymbol{f}_2, \qquad \text{(1-й порядок)}$$

$$s=1: \quad \boldsymbol{y}_2 = \boldsymbol{y}_1 + h(\tfrac{1}{2}\boldsymbol{f}_2 + \tfrac{1}{2}\boldsymbol{f}_1), \qquad \text{(2-й порядок)}$$

$$s=2: \quad \boldsymbol{y}_3 = \boldsymbol{y}_2 + h(\tfrac{5}{12}\boldsymbol{f}_3 + \tfrac{8}{12}\boldsymbol{f}_2 - \tfrac{1}{12}\boldsymbol{f}_1), \qquad \text{(3-й порядок)}$$

$$s=3: \quad \boldsymbol{y}_4 = \boldsymbol{y}_3 + h(\tfrac{9}{24}\boldsymbol{f}_4 + \tfrac{19}{24}\boldsymbol{f}_3 - \tfrac{5}{24}\boldsymbol{f}_2 + \tfrac{1}{24}\boldsymbol{f}_1). \qquad \text{(4-й порядок)}$$

Для методов Адамса справедлива следующая теорема[53].

Теорема. *Для явных методов Адамса $p = s$. Для неявных методов Адамса $p = s + 1$. В обоих случаях порядок метода совпадает с числом точек, по которым строится полиномиальная аппроксимация функции правой части уравнений.*

Для вычисления $\boldsymbol{y}_{k+1}$ в неявных методах приходится решать систему уравнений, которая может оказаться нелинейной. Запишем эту систему уравнений в виде

$$\boldsymbol{y}_{k+1} = \boldsymbol{F}(\boldsymbol{y}_{k+1}),$$

где $\boldsymbol{F}$ – это некоторая функция, см. формулу (15). Для ее решения обычно применяется такой подход: сначала $\boldsymbol{y}_{k+1}$ вычисляется по явному методу, эта оценка обозначается как $\boldsymbol{y}_{k+1}^{(0)}$, а затем методом простой итерации эту оценку уточняют:

$$\boldsymbol{y}_{k+1}^{(m)} = \boldsymbol{F}(\boldsymbol{y}_{k+1}^{(m-1)}).$$

Эта схема получения решения $\boldsymbol{y}_{k+1}$ неявного метода называется «предиктор-корректором». Явный метод, выдающий $\boldsymbol{y}_{k+1}^{(0)}$, называется *пре-*

[53] Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи. Москва : Мир, 1990. С. 336–340 (теорема 2.4, примеры 2.5, 2.6, 2.7 и таблица 2.1).

диктором, а неявный метод – *корректором*. Если шаг h достаточно мал[54], то $\boldsymbol{y}_{k+1}^{(m)}$ сходится к решению неявного метода при $m \to \infty$. На практике обычно ограничиваются одной-двумя итерациями метода простой итерации.

Стартовые значения $\boldsymbol{y}_k, \ldots, \boldsymbol{y}_{k+1-s}$ можно получать, используя либо одношаговые методы, либо методы Адамса более низкого порядка. Так, для расчета $\boldsymbol{y}_2$ можно использовать, например, явный метод Адамса с $s = 1$, для расчета $\boldsymbol{y}_3$ уже можно использовать метод Адамса с $s = 2$ и так далее. Если для расчета стартовых значений используются методы Адамса более низкого порядка, то желательно выбрать для них шаг поменьше, чем планируется для основного метода. Это приводит к необходимости воспользоваться методами Адамса переменного шага, прочитать о них можно в монографии[55].

Управление шагом строится фундаментально на том же самом принципе, что управление шагом во вложенных методах Рунге – Кутты – локальная погрешность оценивается как разность приближенных решений, выдаваемых методами разных порядков. Однако, если в методах Рунге – Кутты для экономии вычислений приходится изобретать методы с одинаковыми стадиями, в методах Адамса это делать не приходится, потому что все эти методы опираются на одни и те же векторы – векторы правых частей в предыдущих узлах. Благодаря тому, что формулы методов Адамса проще, чем у методов Рунге – Кутты, здесь появляется возможность управлять не только шагом, но и порядком метода. На каждом шаге на основании оценки локальной погрешности в соответствии с некоторой логикой можно принимать решения как об изменении шага, так и об изменении порядка рабочего метода Адамса. Это – *методы Адамса с адаптивным шагом переменного порядка*. О них можно прочитать в монографиях[56,57]. Известно несколько реализаций таких методов. Один из таких методов реализован в MATLAB под названием функции `ode113`, другой – как класс `scipy.integrate.LSODA` в пакете `scipy` на языке Python. Этот метод также реализован на Fortran автором в библиотеке KIAM Astrodynamics Toolbox и используется при распространении траекторий космических аппаратов.

---

[54] Холл Дж., Уатт Дж. Современные численные методы решения обыкновенных дифференциальных уравнений. Москва : Мир, 1979. С. 16–17.

[55] Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи. Москва : Мир, 1990.

[56] Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи. Москва : Мир, 1990.

[57] Холл Дж., Уатт Дж. Современные численные методы решения обыкновенных дифференциальных уравнений. Москва : Мир, 1979.

6.8. Сходимость методов Адамса

Методы Адамса являются частными случаями так называемых *линейных многошаговых методов*:

$$\alpha_s \boldsymbol{y}_{k+s} + \alpha_{s-1}\boldsymbol{y}_{k+s-1} + \cdots + \alpha_0 \boldsymbol{y}_k = h(\beta_s \boldsymbol{f}_{n+s} + \cdots + \beta_0 \boldsymbol{f}_n).$$

Для методов Адамса $\alpha_s = 1$, $\alpha_{s-1} = -1$, а остальные α-коэффициенты равны нулю.

Понятия *сходимости* и *порядка сходимости* линейных многошаговых методов определяются для тех задач Коши, для которых решение существует и единственно. Кроме того, предположим, что функция правых частей $\boldsymbol{f}$ непрерывна на множестве $D = \{(t, \boldsymbol{y}) : t \in [t_0, t_f],\ \|\boldsymbol{y}(t) - \boldsymbol{y}\| \le b\}$, где $\boldsymbol{y}(t)$ – точное решение задачи, а b – некоторое положительное число. Будем также предполагать, что функция $\boldsymbol{f}$ на множестве D обладает условием липшицевости с константой Липшица L.

В узлах равномерной с шагом h сетки на отрезке $[t_0, t_f]$ определим решение метода как функцию $\boldsymbol{y}_h(t)$, подчеркивая ее численную природу индексом h. Линейный многошаговый метод называется *сходящимся*, если для всех задач Коши с предположениями выше имеет место сходимость

$$\|\boldsymbol{y}(t) - \boldsymbol{y}_h(t)\| \to 0,\ h \to 0,\ t \in [t_0, t_f]$$

при условии, что для стартовых значений

$$\|\boldsymbol{y}(t_0 + ih) - \boldsymbol{y}_h(t_0 + ih)\| \to 0,\ h \to 0,\ t \in [t_0, t_f],\ i = 0, \ldots, k-1.$$

Говорят, что метод *сходится с порядком* p, если для всех задач Коши со свойством выше, достаточно гладкой функции $\boldsymbol{f}$ и достаточно малого шага h_0 имеют место неравенства

$$\|\boldsymbol{y}(t) - \boldsymbol{y}_h(t)\| \le Ch^{p+1},\ h \le h_0,$$

$$\|\boldsymbol{y}(t_0 + ih) - \boldsymbol{y}_h(t_0 + ih)\| \le C_0 h^{p+1},\ h \le h_0,\ i = 0, \ldots, k-1.$$

Справедлива следующая теорема о методах Адамса[58].

Теорема. *Любой метод Адамса (явный или неявный) с постоянным шагом и порядка p сходится с порядком p и имеет глобальную ошибку (невязку между приближенным и точным значением в момент t_f), равную $O(h^p)$.*

[58] Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи. Москва : Мир, 1990. С. 347, 363.

Известны условия сходимости и для методов с переменным шагом, в них накладываются дополнительные условия на отношение h_k/h_{k-1} длин шагов от итерации к итерации (отношение должно лежать в определенных пределах, зависящих от коэффициентов линейного многошагового метода). Об этом можно прочитать в упомянутой выше монографии, а также в статье[59].

6.9. Производные решений ОДУ по параметрам

Пусть дана система ОДУ вида

$$\begin{cases} \dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{p}), \\ \boldsymbol{y}(t_0) = \boldsymbol{y}_0, \end{cases}$$

где $\boldsymbol{y} \in \mathbb{R}^n$, $t \in [t_0, t_f]$, а $\boldsymbol{p} \in \mathbb{R}^m$ – вектор параметров.

На вопрос о том, как решение этой системы уравнений зависит от параметра $\boldsymbol{p}$, отвечает следующая теорема.

Теорема (Гронуолл)[60]. *Предположим, что при $t_0 \le t \le t_f$ частные производные $\partial \boldsymbol{f}/\partial \boldsymbol{y}$ и $\partial \boldsymbol{f}/\partial \boldsymbol{p}$ существуют и являются непрерывными в окрестности решения $\boldsymbol{y}(t)$. Тогда существуют частные производные $\partial \boldsymbol{y}/\partial \boldsymbol{p}$, причем они непрерывны и удовлетворяют дифференциальному уравнению*

$$\frac{d}{dt}\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{p}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(t, \boldsymbol{y}(t), \boldsymbol{p})\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}(t, \boldsymbol{y}(t), \boldsymbol{p}), \ \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{p}}(t_0) = \boldsymbol{O}_{n\times m}.$$

Теперь допустим, что дана система уравнений

$$\begin{cases} \dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}), \\ \boldsymbol{y}(t_0) = \boldsymbol{y}_0, \end{cases}$$

и нас интересует зависимость решения от начальных условий, то есть от вектора $\boldsymbol{y}_0$ и времени t_0. Обозначим через $\boldsymbol{y}(t, t_0, \boldsymbol{y}_0)$ решение в точке t, удовлетворяющее начальным значениям $\boldsymbol{y}(t_0) = \boldsymbol{y}_0$. Справедлива следующая теорема.

Теорема[61]. *Предположим, что частная производная $\partial \boldsymbol{f}/\partial \boldsymbol{y}$ существует и непрерывна. Тогда решения $\boldsymbol{y}(t, t_0, \boldsymbol{y}_0)$ дифференцируемы*

[59] Shampine L.F., Variable order Adams codes // Computers & Mathematics with Applications. 2002. Vol. 44, N, 5–6. P. 749–761.

[60] Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи. Москва : Мир, 1990. С. 103.

[61] Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи. Москва : Мир, 1990. С. 105–106.

по t_0 *и* $\boldsymbol{y}_0$*, причем* $\partial \boldsymbol{y}/\partial \boldsymbol{y_0}$ *удовлетворяет системе ОДУ*

$$\frac{d}{dt}\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{y}_0} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(t, \boldsymbol{y}(t), \boldsymbol{p})\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{y}_0},\ \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{y}_0}(t_0) = \boldsymbol{E}_n,$$

а производная $\partial \boldsymbol{y}/\partial t_0$ *равна*

$$\frac{\partial \boldsymbol{y}}{\partial t_0}(t) = -\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{y}_0}(t)\boldsymbol{f}(t_0, \boldsymbol{y}_0).$$

Обобщим приведенные выше результаты. Рассмотрим систему ОДУ вида

$$\begin{cases} \dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{p}), \\ \boldsymbol{y}(t_0(\boldsymbol{p})) = \boldsymbol{y}_0(\boldsymbol{p}), \end{cases}$$

где $\boldsymbol{y} \in \mathbb{R}^n$, $t \in [t_0(\boldsymbol{p}), t_f(\boldsymbol{p})]$, то есть пусть теперь и функция правых частей, и начальные и конечные моменты времени зависят от параметра $\boldsymbol{p}$. Наша цель – получить дифференциальные уравнения для функции $\partial \boldsymbol{y}/\partial \boldsymbol{p}$. Как и в теоремах выше, будем считать, что функция $\boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{p})$ непрерывно дифференцируема по t, $\boldsymbol{y}$ и $\boldsymbol{p}$ в окрестности решения, функции $\boldsymbol{y}_0(\boldsymbol{p})$, $t_0(\boldsymbol{p})$, $t_f(\boldsymbol{p})$ непрерывно дифференцируемы по $\boldsymbol{p}$. Обозначим за $\boldsymbol{y}(t, \boldsymbol{p})$ решение, удовлетворяющее начальному условию $\boldsymbol{y}(t_0(\boldsymbol{p})) = \boldsymbol{y}_0(\boldsymbol{p})$.

При указанных условиях из теорем выше следует, что решение $\boldsymbol{y}(t, \boldsymbol{p})$ дифференцируемо по $\boldsymbol{p}$. Действительно, за $\boldsymbol{Y} = \boldsymbol{Y}(t, t_0, \boldsymbol{y}_0, \boldsymbol{p})$ обозначим решение исходных дифференциальных уравнений, отвечающее начальным условиям t_0 и $\boldsymbol{y}_0$ (не связанных с $\boldsymbol{p}$). Из указанных условий и из теорем выше следует, что эта функция непрерывно дифференцируема по всем своим аргументам. Значит, $\boldsymbol{y}(t, \boldsymbol{p}) = \boldsymbol{Y}(t, t_0(\boldsymbol{p}), \boldsymbol{y}_0(\boldsymbol{p}), \boldsymbol{p})$ непрерывно дифференцируема по $\boldsymbol{p}$ и

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{p}} = \frac{\partial \boldsymbol{Y}}{\partial t_0}\frac{\partial t_0}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{y}_0}\frac{\partial \boldsymbol{y}_0}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{p}}.$$

Из теорем выше также следует, что функции справа дифференцируемы по t, значит, и функция слева дифференцируема по t. Вычислим эту производную:

$$\frac{d}{dt}\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{p}} = \left(\frac{d}{dt}\frac{\partial \boldsymbol{Y}}{\partial t_0}\right)\frac{\partial t_0}{\partial \boldsymbol{p}} + \left(\frac{d}{dt}\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{y}_0}\right)\frac{\partial \boldsymbol{y}_0}{\partial \boldsymbol{p}} + \frac{d}{dt}\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{p}} =$$

$$= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\frac{\partial \boldsymbol{Y}}{\partial t_0}\frac{\partial t_0}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{y}_0}\frac{\partial \boldsymbol{y}_0}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} =$$

$$= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\left(\frac{\partial \boldsymbol{Y}}{\partial t_0}\frac{\partial t_0}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{y}_0}\frac{\partial \boldsymbol{y}_0}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{p}}\right) + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}.$$

Что касается начального условия, достаточно продифференцировать равенство $\boldsymbol{y}(t_0(\boldsymbol{p}), \boldsymbol{p}) = \boldsymbol{y}_0(\boldsymbol{p})$ и получить

$$\frac{\partial \boldsymbol{y}_0}{\partial \boldsymbol{p}} = \left.\frac{\partial \boldsymbol{y}}{\partial t}\right|_{t_0(\boldsymbol{p})} \frac{\partial t_0}{\partial \boldsymbol{p}} + \left.\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{p}}\right|_{t_0(\boldsymbol{p})}.$$

Итак, производная $\partial \boldsymbol{y}/\partial \boldsymbol{p}$ является решением следующей задачи Коши:

$$\boxed{\begin{gathered}\frac{d}{dt}\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{p}}(t) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(t, \boldsymbol{y}(t,\boldsymbol{p}))\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{p}}(t) + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}(t, \boldsymbol{y}(t,\boldsymbol{p})), \\ \left.\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{p}}\right|_{t=t_0(\boldsymbol{p})} = \frac{\partial \boldsymbol{y}_0}{\partial \boldsymbol{p}} - \boldsymbol{f}(t_0(\boldsymbol{p}), \boldsymbol{y}_0(\boldsymbol{p}), \boldsymbol{p})\frac{\partial t_0}{\partial \boldsymbol{p}}.\end{gathered}}$$

Заметим, что в эти уравнения входит решение $\boldsymbol{y}(t, \boldsymbol{p})$ исходной системы уравнений. На практике эти уравнения интегрируют совместно с уравнениями исходной системы уравнений. В результате получается $n + nm$ уравнений, которые дают функции $\boldsymbol{y}(t, \boldsymbol{p})$ и $\partial \boldsymbol{y}(t)/\partial \boldsymbol{p}$ для $t \in [t_0(\boldsymbol{p}), t_f(\boldsymbol{p})]$. Важно понимать, что производная $\partial \boldsymbol{y}(t)/\partial \boldsymbol{p}$ в момент $t_f(\boldsymbol{p})$ и производная $\partial \boldsymbol{y}(t_f(\boldsymbol{p}))/\partial \boldsymbol{p}$ не равны, так как в первом случае вычисляется производная по $\boldsymbol{p}$, в которую потом подставляется $t = t_f(\boldsymbol{p})$, а во втором случае сначала берется $\boldsymbol{y}(t_f(\boldsymbol{p}), \boldsymbol{p})$, а затем вычисляется производная по $\boldsymbol{p}$. На самом деле

$$\frac{\partial \boldsymbol{y}(t_f(\boldsymbol{p}), \boldsymbol{p})}{\partial \boldsymbol{p}} = \left.\frac{\partial \boldsymbol{y}}{\partial t}\right|_{t=t_f(\boldsymbol{p})} \frac{\partial t_f}{\partial \boldsymbol{p}} + \left.\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{p}}\right|_{t=t_f(\boldsymbol{p})}.$$

6.10. Интегрирование с выходом на ограничение

В механике космического полета часто встречаются задачи интегрирования уравнений движения с разрывной или негладкой правой частью. Представим, например, что космический аппарат движется под действием двигателя малой тяги, который получает энергию от солнечного излучения с солнечных панелей. В тени Земли двигатель работать не будет. Поэтому уравнения движения в тени и вне нее отличаются на ускорение реактивной тяги. Чтобы корректно проинтегрировать такую систему уравнений, нужно, чтобы метод интегрирования корректно выходил на границу, разделяющую два разных вида движения. Проблема только в том, что заранее может быть не известен момент времени, который соответствует «переключению» правой части с одной на другую. Нужна специальная процедура, позволяющая выйти на

границу, разделяющую два вида движения, – процедура *выхода на ограничение*.

Пусть дана система ОДУ вида

$$\begin{cases} \dot{\boldsymbol{y}}_L = \boldsymbol{f}_L(t, \boldsymbol{y}_L), \\ \boldsymbol{y}_L(t_0) = \boldsymbol{y}_0. \end{cases}$$

Введем скалярную функцию $g = g(t, \boldsymbol{y})$, которую будем называть *функцией события* (*event function*). Будем говорить, что в момент t^* наступает *событие*, если

$$g(t^*, \boldsymbol{y}_L(t^*)) = 0.$$

Теперь пусть дана система ОДУ вида

$$\begin{cases} \dot{\boldsymbol{y}}_R = \boldsymbol{f}_R(t, \boldsymbol{y}_R), \\ \boldsymbol{y}_R(t^*) = \boldsymbol{y}_L(t^*), \end{cases}$$

с, возможно, другой правой частью уравнений $\boldsymbol{f}_R$. Нас интересуют условия, при которых решение $\boldsymbol{y}_R(t_f)$ в конечный момент времени непрерывно зависит от начального условия $\boldsymbol{y}_L(t_0)$, момент времени события t^* и условия сохранения порядка точности численного решения при переходе через момент времени события.

Выясним, при каких условиях решение в конце интервала интегрирования непрерывно зависит от начального условия. Заметим сначала, что решение $\boldsymbol{y}_L = \boldsymbol{y}_L(t, \boldsymbol{y}_0)$ является функцией начального условия, причем при определенных условиях она непрерывно зависит от $\boldsymbol{y}_0$ (см. предыдущий параграф). Момент времени события определяется из уравнения

$$g(t, \boldsymbol{y}_L(t, \boldsymbol{y}_0)) = 0$$

и поэтому тоже зависит от $\boldsymbol{y}_0$. При каких условиях на функцию g эта зависимость тоже является непрерывной? Ответ на это дает теорема о неявной функции.

Теорема[62]. *Пусть функция $F(\boldsymbol{x}, y)$ непрерывна в некоторой окрестности точки $(\boldsymbol{x}_0, y_0)$ и имеет в этой окрестности частную производную F_y, непрерывную в точке $(\boldsymbol{x}_0, y_0)$.*

Если $F(\boldsymbol{x}_0, y_0) = 0$, а $F_y(\boldsymbol{x}_0, y_0) \neq 0$, то найдутся такие окрестности $U(\boldsymbol{x}_0)$, $U(y_0)$, что для каждого $x \in U(\boldsymbol{x}_0)$ существует, и притом единственное, решение $y = f(\boldsymbol{x}) \in U(y_0)$ уравнения $F(\boldsymbol{x}, y) = 0$, причем это решение $y = f(\boldsymbol{x})$ непрерывно на $U(\boldsymbol{x}_0)$ и $y_0 = f(\boldsymbol{x}_0)$.

[62] Кудрявцев Л.Д. Курс математического анализа. В 3 т. Том 2 : учебник для бакалавров. 6 изд., перераб. и доп. Москва : Юрайт, 2014. С. 315.

Если, кроме того, в некоторой окрестности точки $(\boldsymbol{x}_0, y_0)$ существуют все частные производные F_{x_i}, непрерывные в точке $(\boldsymbol{x}_0, y_0)$, то в точке $\boldsymbol{x}_0$ существуют и частные производные f_{x_i}, причем если частные производные F_{x_i} и F_y непрерывны в окрестности точки $(\boldsymbol{x}_0, y_0)$, то частные производные f_{x_i} существуют и непрерывны в некоторой окрестности точки $(\boldsymbol{x}_0, y_0)$.

В нашем случае мы имеем дело с функцией

$$F(\boldsymbol{y}_0, t) \doteq g(t, \boldsymbol{y}_L(t, \boldsymbol{y}_0)).$$

Если функция g непрерывно дифференцируема по обоим своим аргументами и решение $\boldsymbol{y}_L$ непрерывно дифференцируемо по начальному условию, то и функция F непрерывно дифференцируема по своим аргументам. Как следует из теоремы, для существования непрерывной зависимости $t^* = t^*(\boldsymbol{y}_0)$ достаточно, чтобы

$$F_t(\boldsymbol{y}_0, t^*) \neq 0 \Leftrightarrow \frac{d}{dt} g(t, \boldsymbol{y}_L(t, \boldsymbol{y}_0))\Big|_{t=t^*} \neq 0.$$

Более того, так как функция F в нашем случае непрерывно дифференцируема по всем аргументам, зависимость $t^* = t^*(\boldsymbol{y}_0)$ тоже будет непрерывно дифференцируемой.

Полезно понимать геометрический смысл неравенства нулю полной производной функции g на решении ОДУ. Если обозначить за $\boldsymbol{x}(t) = (t, \boldsymbol{y}_L(t, \boldsymbol{y}_0))$ кривую в расширенном фазовом пространстве, то

$$\frac{d}{dt} g(\boldsymbol{x}(t)) \neq 0 \Leftrightarrow \frac{\partial}{\partial \boldsymbol{x}} g(\boldsymbol{x}) \dot{\boldsymbol{x}} \neq 0.$$

Вектор $\dot{\boldsymbol{x}}$ направлен вдоль кривой решения, а $\partial g / \partial \boldsymbol{x}$ направлен ортогонально поверхности $g(\boldsymbol{x}) = 0$. Получается, что кривая решения не должна быть ортогональна вектору, ортогональному поверхности $g(\boldsymbol{x}) = 0$, то есть она не должна быть касательной к этой поверхности. Этот результат интуитивно понятен, так как если бы кривая решения касалась поверхности функции событий, то вариация начального условия могла бы привести к отсутствию пересечения этой поверхности и исчезновению или скачкообразному изменению момента времени события.

Поиск момента t^* можно формализовать как решение некоторого алгебраического уравнения. Численный метод решения ОДУ (например, метод Рунге – Кутты или метод Адамса) выдает последовательность узлов t_1, t_2, t_3, ... и приближенных значений решения в этих узлах $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, $\boldsymbol{y}_3$, ... Поэтому на каждой итерации интегратора можно сделить за

знаком числа $g(t_k, \boldsymbol{y}_k)$ и остановиться на моменте, когда знак меняется, то есть остановиться на k таком, что

$$g(t_k, \boldsymbol{y}_k) g(t_{k+1}, \boldsymbol{y}_{k+1}) < 0.$$

Если функция событий непрерывная, то существует точка $t^* \in [t_k, t_{k+1}]$ и шаг $h^* = t^* - t_k$, для которых $g(t^*, \boldsymbol{y}_h^*) = 0$, где $\boldsymbol{y}_h^*$ – численное решение интегратора, которое получается на одном шаге величины h^* от точки t_k. Остается только найти этот шаг h^*, то есть решить скалярное уравнение

$$\varphi(h) \doteq g(t_k + h, \boldsymbol{y}_h^*) = 0$$

относительно скалярной переменной h на отрезке $[0, t_{k+1} - t_k]$. Для этого можно воспользоваться методом дихотомии или методом Ньютона. О точности определения времени события и ее связи с глобальной ошибкой интегрирования можно прочитать в статье[63]. Для решения уравнения можно воспользоваться и методами интерполяции. Например, если g в окрестности события меняется строго монотонно, то по значениям вектор-функции $\{(t_k, \boldsymbol{y}_k)\}$ в точках $\{g_k = g(t_k, \boldsymbol{y}_k)\}$ можно построить интерполяционный полином и искать значение этого полинома в точке $g = 0$. В более общем случае по $\{(t_k, \boldsymbol{y}_k)\}$ строится полином, и численными итерационными методами ищется его аргумент, который дает $g = 0$. Многошаговые методы наиболее удобны в случаях, когда для поиска события используются интерполяционные полиномы, так как они сами представляют решение в виде полинома.

После того как t^* найдено с достаточной точностью (не меньше, чем точность интегратора), интегратор запускается из начального условия $\boldsymbol{y}_h^*$ в момент t^* для правой части $\boldsymbol{f}_R$. Естественно, описанные выше подходы не гарантируют нахождение t^*, например, потому что шаг интегратора может быть слишком большим, чтобы определить смену знака функции g. Максимальный размер шага интегрирования следует установить пользователю самостоятельно исходя из физических аспектов рассматриваемой задачи. Например, если аппарат должен пролететь тень, то шаг интегратора должен быть достаточно малым, чтобы хотя бы одна итерация интегратора попала внутрь тени, а это значит, что нужно связать скорость аппарата и размеры тени. Помочь не пропустить возможную точку события помогает контроль не только знака функции g, но и знака производной g по времени.

[63] Mannshardt R. One-step methods of any order for ordinary differential equations with discontinuous right-hand sides // Numerische Mathematik. 1978. Vol. 31, N. 2. P. 131–152.

6.11. Методы решения ОДУ с ограничениями

В механике космического полета бывает необходимо решить задачу не только с начальными условиями, но и с краевыми условиями, и вообще с произвольными ограничениями на векторы зависимых переменных. Проще всего пояснить методы решения таких задач на примерах.

Рассмотрим задачу проектирования перелета из одной точки в другую в возмущенном поле некоторого притягивающего центра. Поместим начало системы координат в притягивающий центр и обозначим радиус-векторы точек, между которыми нужно найти траекторию перелета, за $\boldsymbol{R}_0$ и $\boldsymbol{R}_f$. Рассмотрим задачу поиска решения дифференциального уравнения с краевыми условиями:

$$\dot{\boldsymbol{r}} = \boldsymbol{v},\ \dot{\boldsymbol{v}} = -\boldsymbol{r}/r^3 + \boldsymbol{F}(t, \boldsymbol{r}, \boldsymbol{v}),\ \boldsymbol{r}(t_0) = \boldsymbol{R}_0,\ \boldsymbol{r}(t_f) = \boldsymbol{R}_f$$

на интервале времени $t \in [t_0, t_f]$. Здесь $\boldsymbol{F}(t, \boldsymbol{r}, \boldsymbol{v})$ – возмущающее ускорение. Если $\boldsymbol{F} \equiv 0$, то указанная краевая задача называется *задачей Ла́мберта*, ее решения нельзя выразить в элементарных функциях, но можно получить численно с помощью эффективных методов[64]. Мы рассмотрим применение метода пристрелки к решению задачи с произвольной функцией $\boldsymbol{F}$.

Суть *метода пристрелки* состоит в том, чтобы свести краевую задачу к последовательности задач Коши. Мы найдем решение краевой задачи выше, если найдем подходящую скорость $\boldsymbol{v}_0$ в начальный момент времени t_0, такую, чтобы интегрирование траектории с начальным условием $\boldsymbol{r}(t_0) = \boldsymbol{R}_0$, $\boldsymbol{v}(t_0) = \boldsymbol{v}_0$ на интервале времени $[t_0, t_f]$, приводило к $\boldsymbol{r}(t_f) = \boldsymbol{R}_f$. Тогда краевая задача будет решена. Это можно формализовать следующим образом. Пусть

$$\boldsymbol{r}(t) = \boldsymbol{r}(t; \boldsymbol{r}_0, \boldsymbol{v}_0),\ \boldsymbol{v}(t) = \boldsymbol{v}(t; \boldsymbol{r}_0, \boldsymbol{v}_0)$$

суть положение и скорость аппарата в момент t, отвечающие начальным условиям $\boldsymbol{r}(t_0) = \boldsymbol{r}_0$ и $\boldsymbol{v}(t_0) = \boldsymbol{v}_0$. Тогда нас интересует решение уравнения

$$\boldsymbol{r}(t_f; \boldsymbol{R}_0, \boldsymbol{v}_0) = \boldsymbol{R}_f$$

относительно переменной $\boldsymbol{v}_0$. Это система из трех нелинейных уравнений с тремя неизвестными, и ее можно решать, например, методом Ньютона. Метод Ньютона будет подбирать вектор $\boldsymbol{v}_0$ до тех пор, пока решение $\boldsymbol{r}(t_f; \boldsymbol{R}_0, \boldsymbol{v}_0)$ задачи Коши в момент t_f с начальным условием

[64] Izzo D.. Revisiting Lambert's Problem. URL: arXiv:1403.2705 (Дата обращения 18.10.2023).

$[\boldsymbol{R}_0,\, \boldsymbol{v}_0]$ не приведет к малой невязке между $\boldsymbol{r}(t_f; \boldsymbol{R}_0, \boldsymbol{v}_0)$ и $\boldsymbol{R}_f$ (перебор начального условия и дальнейшее распространение траектории интерпретируется как «пристрелка»).

Как известно, методу Ньютона требуется производить расчет производной функции невязки по переменной. В нашем случае ему потребуется матрица

$$\frac{\partial \boldsymbol{r}(t_f; \boldsymbol{R}_0, \boldsymbol{v}_0)}{\partial \boldsymbol{v}_0},$$

то есть производная радиус-вектора в конце интервала интегрирования по начальной скорости. Эта матрица является частью матрицы производных решения по начальному условию

$$\frac{\partial \boldsymbol{y}(t_f)}{\partial \boldsymbol{y}_0} = \begin{bmatrix} \dfrac{\partial \boldsymbol{r}(t_f)}{\partial \boldsymbol{r}_0} & \dfrac{\partial \boldsymbol{r}(t_f)}{\partial \boldsymbol{v}_0} \\ \dfrac{\partial \boldsymbol{v}(t_f)}{\partial \boldsymbol{r}_0} & \dfrac{\partial \boldsymbol{v}(t_f)}{\partial \boldsymbol{v}_0} \end{bmatrix},$$

где $\boldsymbol{y} = [\boldsymbol{r}, \boldsymbol{v}]$. Эта матрица, в свою очередь, может быть получена как решение уравнений в вариациях, о которых шла речь ранее.

Формализовать метод пристрелки как решение некоторого алгебраического уравнения можно по-разному. Например, неизвестной переменной можно считать вектор $[\boldsymbol{r}_0, \boldsymbol{v}_0]$ с шестью компонентами. Тогда система уравнений запишется так:

$$\boldsymbol{r}(t_f; \boldsymbol{r}_0, \boldsymbol{v}_0) = \boldsymbol{R}_f,\ \ \boldsymbol{r}_0 = \boldsymbol{R}_0.$$

В этой системе присутствует уже шесть уравнений на шесть неизвестных. Второе уравнение тривиально разрешается, его решение подставляется в первое уравнение, и мы приходим к системе уравнений, выписанной ранее. В принципе, эта расширенная система уравнений в данном случае не нужна, но допустим, что на начальную точку выставлены более общие условия, например, в начальный момент должно быть $\boldsymbol{g}(\boldsymbol{r}_0, \boldsymbol{v}_0) = 0$ для некоторой функции $\boldsymbol{g}$. Тогда можно решать систему уравнений, записанную в общей форме

$$\boldsymbol{r}(t_f; \boldsymbol{r}_0, \boldsymbol{v}_0) = \boldsymbol{R}_f,\ \ \boldsymbol{g}(\boldsymbol{r}_0, \boldsymbol{v}_0) = 0.$$

Конечно, ни исходная краевая задача, ни эти алгебраические уравнения не обязаны иметь решение (и даже если имеют, оно не обязано быть единственным). Отсутствие решения может быть одной из причин несходимости метода Ньтона или любого другого метода.

Теперь рассмотрим задачу перелета с орбиты вокруг Земли на орбиту вокруг Луны. Допустим, что аппарат движется вокруг Земли по некоторой орбите и требуется найти начальные условия, которые бы перевели его с этой орбиты на выбранную орбиту вокруг Луны. Поставим краевую задачу:

$$\dot{\boldsymbol{r}} = \boldsymbol{v},\ \dot{\boldsymbol{v}} = -\boldsymbol{r}/r^3 + \boldsymbol{F}(t, \boldsymbol{r}, \boldsymbol{v}),\ \boldsymbol{g}_0(\boldsymbol{r}(t_0), \boldsymbol{v}(t_0)) = 0,\ \boldsymbol{g}_f(\boldsymbol{r}(t_f), \boldsymbol{v}(t_f)) = 0,$$

на интервале времени $t \in [t_0, t_f]$, где $\boldsymbol{g}_0$ и $\boldsymbol{g}_f$ – известные исследователю функции, которые определяют ограничения на начальные и конечные условия. Для того чтобы решить эту краевую задачу, можно ввести переменные $\boldsymbol{r}_0$ и $\boldsymbol{v}_0$ положения и скорости аппарата в начальный момент времени и составить систему уравнений:

$$\boldsymbol{g}_0(\boldsymbol{r}_0, \boldsymbol{v}_0) = 0,\ \boldsymbol{g}_f(\boldsymbol{r}(t_f; \boldsymbol{r}_0, \boldsymbol{v}_0), \boldsymbol{v}(t_f; \boldsymbol{r}_0, \boldsymbol{v}_0)) = 0.$$

Можно ввести переменные $\boldsymbol{r}_0$, $\boldsymbol{v}_0$, $\boldsymbol{r}_f$, $\boldsymbol{v}_f$ положений и скоростей аппарата в начальный и конечный моменты времени и составить более «симметричную» систему уравнений:

$$\boldsymbol{g}_0(\boldsymbol{r}_0, \boldsymbol{v}_0) = 0,\ \boldsymbol{g}_f(\boldsymbol{r}_f, \boldsymbol{v}_f) = 0,$$

$$\boldsymbol{r}(t_f; \boldsymbol{r}_0, \boldsymbol{v}_0) = \boldsymbol{r}_f,\ \boldsymbol{v}(t_f; \boldsymbol{r}_0, \boldsymbol{v}_0) = \boldsymbol{v}_f.$$

В зависимости от ситуации эти системы могут не иметь решения, иметь единственное решение или иметь множество решений (в том числе и бесконечное). Число уравнений может не совпадать с числом переменных, все зависит от накладываемых ограничений. В случаях, когда существует целое множество решений, разумно ограничиться поиском только оптимального в каком-либо смысле решения. Например, можно искать решение, оптимальное по затратам характеристической скорости в начальный и конечный моменты времени, то есть решать оптимизационную задачу:

$$|\boldsymbol{v}_0 - \boldsymbol{V}_0|^2 + |\boldsymbol{v}_f - \boldsymbol{V}_f|^2 \to \min,$$

$$\boldsymbol{g}_0(\boldsymbol{r}_0, \boldsymbol{v}_0) = 0,\ \boldsymbol{g}_f(\boldsymbol{r}_f, \boldsymbol{v}_f) = 0,$$

$$\boldsymbol{r}(t_f; \boldsymbol{r}_0, \boldsymbol{v}_0) = \boldsymbol{r}_f,\ \boldsymbol{v}(t_f; \boldsymbol{r}_0, \boldsymbol{v}_0) = \boldsymbol{v}_f.$$

где $\boldsymbol{V}_0$ и $\boldsymbol{V}_f$ – скорости аппарата на околоземной орбите до импульса скорости и на окололунной орбите после импульса скорости. Эти скорости могут зависеть от точки старта и соответственно от $\boldsymbol{r}_0$. Отмечу пользу от введения, казалось бы, избыточных переменных $\boldsymbol{r}_f$ и $\boldsymbol{v}_f$.

Если этого не делать и заменить в целевой функции $\boldsymbol{v}_f$ на $\boldsymbol{v}(t_f; \boldsymbol{r}_0, \boldsymbol{v}_0)$, то при расчете производной целевой функции по $\boldsymbol{r}_0$ и $\boldsymbol{v}_0$ понадобится производная фазового вектора по начальной скорости, которую необходимо рассчитывать численно и которая будет использоваться и в производной целевой функции, и в производной функций-ограничений. Если же вводится независимая переменная $\boldsymbol{v}_f$, то целевая функция будет просто квадратичной функцией переменных $\boldsymbol{v}_0$ и $\boldsymbol{v}_f$ с аналитически вычисляемой точной производной и положительно определенной матрицей вторых производных. Распространение траекторий и подсчет производных конечных фазовых векторов по начальным условиям производится только в функциях ограничений, но не в целевой функции. Это упрощает программную реализацию метода.

В оптимизационную задачу можно добавить и любые другие ограничения, в том числе и ограничения-неравенства, и не только в краевых точках, но и в любых промежуточных точках. Моменты времени t_0 и t_f можно тоже причислить к переменным и подвергнуть их оптимизации.

Особенностью указанных выше алгебраических уравнений и оптимизационных задач является то, что в них входит решение некоторых ОДУ, а переменные входят в их начальные условия. Иногда, если интервалы интерирования достаточно велики, а динамика аппарата сложна, может потребоваться очень хорошее начальное приближение к оптимальному значению переменных для того, чтобы методы решения уравнений и методы оптимизации сошлись к искомому решению. Траектория в таких случаях очень чувствительна к переменным оптимизации.

Метод параллельной пристрелки – один из самых эффективных способов справиться с проблемой высокой чувствительности решений ОДУ к начальным условиям при решении краевых задач и вообще оптимизационных задач, в которых приходится интегрировать траектории аппарата. Его суть состоит в том, чтобы рассматривать не один интервал интегрирования, а серию интервалов, интегрировать уравнения движения на этих интервалах независимо друг от друга и «сшивать» эти части решений на соседних интервалах.

Рассмотрим, к примеру, задачу оптимизации траектории перелета между околоземной и окололунной орбитами, описанную выше. В ограничениях-равенствах присутствуют векторы $\boldsymbol{r}(t_f; \boldsymbol{r}_0, \boldsymbol{v}_0)$ и $\boldsymbol{v}(t_f; \boldsymbol{r}_0, \boldsymbol{v}_0)$, которые получаются в результате интегрирования траектории перелета на всем интервале $[t_0, t_f]$. Пусть, как и прежде, $\boldsymbol{y} = [\boldsymbol{r}, \boldsymbol{v}]$. Введем моменты времени

$$t_0 \le t_1 < t_2 < \cdots < t_{N-1} < t_N \le t_f$$

и переменные оптимизации

$$\boldsymbol{y}_0,\ \boldsymbol{y}_1,\ \boldsymbol{y}_2,\ \ldots, \boldsymbol{y}_{N-1},\ \boldsymbol{y}_N,\ \boldsymbol{y}_f.$$

Тогда новую задачу оптимизацию можно записать так:

$$|\boldsymbol{v}_0 - \boldsymbol{V}_0|^2 + |\boldsymbol{v}_f - \boldsymbol{V}_f|^2 \to \min,$$

$$\boldsymbol{g}_0(\boldsymbol{y}_0) = 0,\ \boldsymbol{g}_f(\boldsymbol{y}_f) = 0,\ \boldsymbol{y}(t_k; \boldsymbol{y}_{k-1}) = \boldsymbol{y}_k,\ \forall k = 1, \ldots, N,\ \boldsymbol{y}(t_f; \boldsymbol{y}_N) = \boldsymbol{y}_f.$$

Здесь целевая функция – это квадрат затрат характеристической скорости на коррекции скорости в начале и конце полета. Условия $\boldsymbol{g}_0(\boldsymbol{y}_0) = 0$ и $\boldsymbol{g}_f(\boldsymbol{y}_f) = 0$ устанавливают ограничения на начальный и конечный фазовые векторы траектории перелета. Остальные условия означают непрерывную сшивку частей траектории на разных интервалах. А именно, $\boldsymbol{y}(t_k; \boldsymbol{y}_{k-1})$ – это результат интегрирования уравнений движения на интервале $[t_{k-1}, t_k]$ с начальным условием $\boldsymbol{y}_{k-1}$. Условие сшивки означает, что этот результат интегрирования должен быть равен переменной оптимизации $\boldsymbol{y}_k$.

Моменты времени t_1, t_2, …, t_N могут быть фиксированными, а могут быть тоже переменными оптимизации. В таком случае в оптимизационную задачу следует добавить условия-неравенства

$$t_0 \le t_1 < t_2 < \cdots < t_{N-1} < t_N \le t_f.$$

В общем случае в оптимизационную задачу могут входить ограничения в каких-то промежуточных точках, в том числе и в t_k, а сшивка не обязательно может быть непрерывная, можно потребовать непрерывность только по положению, а разрыв по скорости будет интерпретироваться как импульс скорости в этой точке, который тоже можно добавить в целевую функцию.

Итак, в общем случае решения ОДУ с ограничениями строятся по одной схеме – ставится некоторая задача оптимизации, для которой вводится серия переменных оптимизации, которые связываются между собой так, как нужно для получения искомого решения. Эти связи выражаются между собой в терминах функций-ограничений равенства или неравенства, а целевая функция используется для поиска и отбора оптимального в каком-либо смысле решения.

7. Уравнения орбитальной механики

7.1. Вспомогательные уравнения

Движение аппарата в возмущенном поле притягивающего центра описывается дифференциальными уравнениями:

$$\dot{\boldsymbol{r}} = \boldsymbol{v},\ \dot{\boldsymbol{v}} = -\mu \boldsymbol{r}/r^3 + \boldsymbol{F}(t, \boldsymbol{r}, \boldsymbol{v}),$$

где μ – гравитационный параметр притягивающего центра, $\boldsymbol{F}(t, \boldsymbol{r}, \boldsymbol{v})$ – ускорение возмущающих сил, которое может быть вызвано сопротивлением атмосферы, несферичностью поля притягивающего центра, гравитационным притяжением к небесным телам, давлением солнечного излучения, реактивным ускорением тяги на аппарате и другими силами. Если $\boldsymbol{F} \equiv 0$, то указанную модель движения называют *моделью задачи двух тел*.

Введем в рассмотрение функции

$$h(\boldsymbol{r}, \boldsymbol{v}) = v^2/2 - \mu/r,\ \boldsymbol{c}(\boldsymbol{r}, \boldsymbol{v}) = \boldsymbol{r} \times \boldsymbol{v},$$

$$\boldsymbol{A}(\boldsymbol{r}, \boldsymbol{v}) = -\mu \boldsymbol{r}/r + \boldsymbol{v} \times (\boldsymbol{r} \times \boldsymbol{v}).$$

Известно, что на решениях задачи двух тел эти функции постоянны, то есть в задаче двух тел они являются интегралами движения. При этом функцию h называют *интегралом энергии*, $\boldsymbol{c}$ – *интегралом площадей*, $\boldsymbol{A}$ – *интегралом Лапласа*.

В возмущенной задаче двух тел, то есть когда $\boldsymbol{F}$ не нулевая функция, эти функции на решениях уравнений движения, вообще говоря, зависят от времени. Продифференцируем эти функции по времени с учетом уравнений движения:

$$\frac{dh}{dt} = \frac{d}{dt}\frac{\boldsymbol{v} \cdot \boldsymbol{v}}{2} - \frac{d}{dt}\frac{\mu}{r} = \dot{\boldsymbol{v}} \cdot \boldsymbol{v} + \frac{\mu \dot{r}}{r^2} = -\mu \frac{\boldsymbol{r} \cdot \boldsymbol{v}}{r^3} + \boldsymbol{v} \cdot \boldsymbol{F} + \mu \frac{\boldsymbol{v} \cdot \boldsymbol{r}}{r^3} = \boldsymbol{v} \cdot \boldsymbol{F}.$$

$$\frac{d\boldsymbol{c}}{dt} = \dot{\boldsymbol{r}} \times \boldsymbol{v} + \boldsymbol{r} \times \dot{\boldsymbol{v}} = \boldsymbol{r} \times \boldsymbol{F}.$$

$$\frac{d\boldsymbol{A}}{dt} = -\mu \frac{d}{dt}\frac{\boldsymbol{r}}{r} + \dot{\boldsymbol{v}} \times \boldsymbol{c} + \boldsymbol{v} \times \dot{\boldsymbol{c}} =$$

$$= -\mu \frac{\boldsymbol{v}}{r} + \mu \frac{\boldsymbol{r}}{r^2}\dot{r} - \mu \frac{\boldsymbol{r} \times \boldsymbol{c}}{r^3} + \boldsymbol{F} \times \boldsymbol{c} + \boldsymbol{v} \times (\boldsymbol{r} \times \boldsymbol{F}) = \boldsymbol{F} \times \boldsymbol{c} + \boldsymbol{v} \times (\boldsymbol{r} \times \boldsymbol{F}).$$

Итак, в результате получаются формулы

$$\frac{dh}{dt} = \boldsymbol{v} \cdot \boldsymbol{F},\ \frac{d\boldsymbol{c}}{dt} = \boldsymbol{r} \times \boldsymbol{F},\ \frac{d\boldsymbol{A}}{dt} = \boldsymbol{F} \times \boldsymbol{c} + \boldsymbol{v} \times (\boldsymbol{r} \times \boldsymbol{F}).$$

Важно заметить, что на эти формулы можно посмотреть как на дифференциальные уравнения, которые можно интегрировать совместно с уравнениями движения аппарата. Решение этих уравнений известно, это выражения для функций h, $\boldsymbol{c}$, $\boldsymbol{A}$, выписанные выше. Может показаться, что вводить дифференциальные уравнения для этих функций вместо того, чтобы вычислять их значения непосредственно – странная затея. Но дальше мы увидим, что в этом есть смысл.

7.2. Сингулярность и неустойчивость уравнений

Уравнения движения, записанные выше, вырождаются при $r = 0$, в связи с чем возникают определенные трудности при интегрировании движения в области, близкой к центру. Значения и производные функции правых частей резко возрастают и изменяются вблизи $r = 0$. Это усиливает требования на шаг интегрирования для сохранения локальной точности. Поэтому интеграторы с адаптивным шагом в области перицентра уменьшают шаг интегрирования, а в области, далекой от притягивающего центра, они его увеличивают. Далее мы рассмотрим методы регуляризации уравнений движения, то есть поиска таких преобразований времени и координат, после которых уравнения движения не имеют особенности при $r = 0$ и будут допускать интегрирование вплоть до точки с $r = 0$ включительно. У интеграторов не будет небходимости уменьшать шаг интегрирования в окрестности $r = 0$, чтобы сохранить точность.

Есть еще одна проблема, связанная с решениями уравнений движения, написанными выше. Дело в том, что они неустойчивы по Ляпунову относительно начальных условий. При эллиптическом движении вариации начальных положения $\delta\boldsymbol{r}_0$ и скорости $\delta\boldsymbol{v}_0$ приводят к вариации энергии $\delta h = v\delta v + \mu\delta r/r^2$, вариации большой полуоси $\delta a = \mu/h^2\delta h$ и, как следствие, периода движения $\delta T = 3\pi\sqrt{a/\mu}\delta a$. Получается, что какими бы малыми не были $\delta\boldsymbol{r}_0$ и $\delta\boldsymbol{v}_0$, всегда найдутся такие их направления, что $\delta T \neq 0$. Точки, движущиеся с отличающимися таким образом начальными условиями, разойдутся на конечное расстояние за конечное время. Что касается параболического и гиперболического движения, достаточно изменить эксцентриситет орбиты. У параболической орбиты достаточно уменьшить эксцентриситет, это превратит неограниченное движение в ограниченное. У гиперболической орбиты изменение эксцентриситета изменит асимптоты к траектории, что приведет к расходимости разности положений. Прямолинейные траектории стремятся либо к центральному телу, либо уходят на бесконечность. Придание нормальной к траектории компоненты приведет к орбите,

которая будет либо эллиптической, либо уйдет на бесконечность вдоль отличной от прямолинейной траектории орбиты и не столкнется с притягивающим центром. Интеграторы определяют решение уравнений движения лишь с определенной точностью, а ошибки в определении положения и скорости могут транслироваться в указанные выше ошибки по большой полуоси или эксцентриситету, что будет вызывать расходимость разности положений аппарата, даже если в какой-то момент ошибки интегратора пропадут.

Оказывается, можно сделать такие замены переменных и времени, чтобы решения уравнений движения в новых переменных в задаче двух тел стали устойчивыми. Приведу показательный пример того, как одна лишь замена времени делает все неустойчивые решения системы устойчивыми. Рассмотрим дифференциальное уравнение

$$\dot{y} = y,\ t \geq 0$$

с $y(0) > 0$. Все решения $y(t) = Ce^t$, $C = y(0)$, этого уравнения, как легко показать, являются неустойчивыми по Ляпунову, то есть

$$\exists \varepsilon > 0\ \forall \delta > 0\ \exists T > 0 :\ |y_2(0) - y_1(0)| < \delta,\quad |y_2(T) - y_1(T)| > \varepsilon.$$

Теперь перейдем к новой независимой переменной s, связанной с t как $dt = y^{-1}ds$. Производную по s будем обозначать штрихом $(\cdot)'$. Уравнение выше перепишется в виде двух уравнений:

$$y' = 1,\ t' = y^{-1}.$$

Эти уравнения имеют решение

$$y(s) = s + y(0),\ t(s) = \ln(s + y(0)) - \ln y(0),$$

если считать $t(0) = 0$. Все эти решения устойчивы по Ляпунову: $|y_2(s) - -y_1(s)| = |y_2(0) - y_1(0)|$ для всех s и

$$|t_2(s) - t_1(s)| = \left|\ln \frac{s + y_2(0)}{s + y_1(0)} - \ln \frac{y_2(0)}{y_1(0)}\right| \leq$$

$$\leq \left|\ln\left(1 + \frac{y_2(0) - y_1(0)}{s + y_1(0)}\right)\right| + \left|\ln\left(1 + \frac{y_2(0) - y_1(0)}{y_1(0)}\right)\right|,$$

откуда, воспользовавшись неравенством $|\ln(1 + x)| \leq (2\ln 2)|x|$ для $|x| \leq 1/2$,

$$|t_2(s) - t_1(s)| \leq (2\ln 2) \cdot \left(\left|\frac{y_2(0) - y_1(0)}{s + y_1(0)}\right| + \left|\frac{y_2(0) - y_1(0)}{y_1(0)}\right|\right) \leq$$

$$\leq (4 \ln 2) \left| \frac{y_2(0) - y_1(0)}{y_1(0)} \right|.$$

Выражение справа может быть сделано сколь угодно малым, поэтому

$$\forall \varepsilon > 0 \ \exists \delta > 0 \ \forall t > 0 : \ |y_2(0) - y_1(0)| < \delta,$$

$$|y_2(s) - y_1(s)| < \varepsilon, \ |t_2(s) - t_1(s)| < \varepsilon.$$

Итак, мы получили, что все решения исходных уравнений неустойчивы и все решения новых уравнений устойчивы, хотя и те, и другие уравнения описывают одну и ту же динамику.

Все дело в том, что после преобразования исходное время $t(s)$ «течет» по-разному для разных решений y_1 и y_2, а именно $y_1(s) = y_1(t_1(s))$, $y_2(s) = y_2(t_2(s))$. Получается, что разность решений в одинаковые моменты t расходится с ростом t, а разность решений в одинаковые моменты s (и как следствие – разные моменты t) сохраняется с ростом s. Причина этого интересного эффекта состоит в том, что в определении устойчивости мы пользуемся всего одним способом определения расстояния между решениями – модулем разности решений при одинаковых значениях независимых переменных. После преобразования времени $t \to s$ разность решений при одинаковых t не совпадает с разностью решений при одинаковых s, так как «течение времени» $t = t(s)$ зависит от начальных условий сравниваемых решений.

Приведу результаты численного интегрирования уравнений выше. Если интегрировать уравнения на участках $t \in [0, 1]$ и $s \in [0, e - 1]$ методом Дормана – Принса 5(4) с локальной абсолютной и относительной точностью `1.0E-03`, то отклонение решений в точке $t = 1$ от истинного значения составит `4.3E-05` для исходных уравнений и `4.4E-16` для стабилизированных уравнений. При этом интегратор делает 14 шагов при интегрировании исходных уравнений и 20 шагов при интегрировании стабилизированных уравнений. Если заменить точность интегрирования исходных уравнений на `1E-4`, то для их интегрирования понадобится тоже 20 шагов, а точность станет `1.9E-05`. Таким образом, интегрирование стабилизированных уравнений позволяет получить решение на много порядков точнее и даже быстрее.

Пример выше показывает, что устойчивость или неустойчивость решений – это не свойство динамики как таковой, а свойство описывающих эту динамику зависимых и независимых переменных. Одна и та же динамика, записанная в разных терминах, может быть как устойчивой, так и неустойчивой. Уравнения движения в задаче двух тел, записанные в терминах $\boldsymbol{r}$ и $\boldsymbol{v}$, дают неустойчивые движения. Оказывается, существуют такие преобразования координат и времени,

которые описывают ту же самую динамику задачи двух тел, но все решения получающихся уравнений являются устойчивыми по Ляпунову. Приступая к выводу таких уравнений, введем основное преобразование времени, которое позволит перейти к устойчивым уравнениям, – *преобразование Сундмана*:

$$dt = r ds.$$

7.3. Уравнения Шперлинга – Боде

Уравнения возмущенного движения в терминах s из преобразования Сундмана записываются в виде

$$\boldsymbol{r}' = r\boldsymbol{v},\ \boldsymbol{v}' = -\mu\boldsymbol{r}/r^2 + r\boldsymbol{F}(t, \boldsymbol{r}, \boldsymbol{v}),\ t' = r.$$

Перейдем к уравнениям второго порядка. Заметим, что

$$r' = \frac{\boldsymbol{r}' \cdot \boldsymbol{r}}{r} = \frac{r\boldsymbol{v} \cdot \boldsymbol{r}}{r} = \boldsymbol{v} \cdot \boldsymbol{r},$$

$$\boldsymbol{r}'' = (r\boldsymbol{v})' = r'\boldsymbol{v} + r\boldsymbol{v}' = (\boldsymbol{v} \cdot \boldsymbol{r})\boldsymbol{v} - \mu\boldsymbol{r}/r + r^2\boldsymbol{F}.$$

Из выражения для вектора Лапласа получаем

$$\boldsymbol{A} = -\mu\boldsymbol{r}/r + \boldsymbol{v} \times (\boldsymbol{r} \times \boldsymbol{v}) = -\mu\boldsymbol{r}/r + \boldsymbol{r}v^2 - \boldsymbol{v}(\boldsymbol{r} \cdot \boldsymbol{v}),$$

$$\boldsymbol{v}(\boldsymbol{r} \cdot \boldsymbol{v}) = -\boldsymbol{A} - \mu\boldsymbol{r}/r + \boldsymbol{r}v^2,$$

и, следовательно,

$$\boldsymbol{r}'' = -\boldsymbol{A} - 2\mu\boldsymbol{r}/r + \boldsymbol{r}v^2 + r^2\boldsymbol{F}.$$

Вспомним также, что $2h = v^2 - 2\mu/r$, поэтому

$$\boldsymbol{r}'' = -\boldsymbol{A} + 2h\boldsymbol{r} + r^2\boldsymbol{F}.$$

Продифференцировав r' по s и действуя аналогичным образом, можно прийти к формуле

$$r'' - 2hr - \mu = r\boldsymbol{F} \cdot \boldsymbol{r}.$$

А теперь проделаем трюк – будем воспринимать $\boldsymbol{A}$ и h как зависимые переменные, удовлетворяющие дифференциальным уравнениям. Тогда мы получим систему уравнений

$$\boldsymbol{r}'' - 2h\boldsymbol{r} + \boldsymbol{A} = r^2\boldsymbol{F},\ r'' - 2hr - \mu = r\boldsymbol{F} \cdot \boldsymbol{r},$$

$$h' = r\boldsymbol{v}\cdot\boldsymbol{F},\ \boldsymbol{A}' = r\boldsymbol{F}\times\boldsymbol{c} + r\boldsymbol{v}\times(\boldsymbol{r}\times\boldsymbol{F}),\ t' = r.$$

Ее можно переписать в виде системы первого порядка:

$$\boldsymbol{r}' = \boldsymbol{w},\ \boldsymbol{w}' = 2h\boldsymbol{r} - \boldsymbol{A} + r^2\boldsymbol{F},$$

$$r' = q,\ q' = 2hr + \mu + r\boldsymbol{F}\cdot\boldsymbol{r},$$

$$h' = \boldsymbol{w}\cdot\boldsymbol{F},\ \boldsymbol{A}' = \boldsymbol{F}\times\boldsymbol{w} + \boldsymbol{w}\times(\boldsymbol{r}\times\boldsymbol{F}),\ t' = r,$$

где $\boldsymbol{w} = r\boldsymbol{v} = \boldsymbol{r}'$ – производная радиус-вектора по переменной s, а $q = r'$ – производная расстояния до притягивающего центра по переменной s. Эти уравнения и называются уравнениями *Шперлинга – Боде* (*Sperling – Burdet*).

Обсудим свойства этих уравнений. Во-первых, эти уравнения не вырождаются при $r = 0$, слагаемые с $1/r$ «спрятаны» в переменные $\boldsymbol{A}$ и h, которые мы теперь не вычисляем по их формулам, а находим из решения этих дифференциальных уравнений. Во-вторых, при $\boldsymbol{F} \equiv 0$, то есть в случае отсутствия возмущений, все уравнения здесь *линейные*:

$$\boldsymbol{r}'' - 2h\boldsymbol{r} + \boldsymbol{A} = 0,\ r'' - 2hr - \mu = 0,$$

$$h' = 0,\ \boldsymbol{A}' = 0,\ t' = r.$$

Замечу, что именно для того, чтобы сделать их линейными, мы ввели отдельно кажущееся ненужным уравнение для r, ведь иначе $r = |\boldsymbol{r}|$ и уравнение $t' = r$ было бы нелинейным. Кроме того, уравнения для $\boldsymbol{r}$ и r – это уравнения гармонического осциллятора. В-третьих, если исключить из этих невозмущенных уравнений уравнения на h и $\boldsymbol{A}$, а задавать их перед интегрированием равными какому-то фиксированному значению, то полученная система уравнений для эллиптического случая будет устойчивая. Покажем это. Приращения $\Delta\boldsymbol{r}$, Δr, Δt решений, отвечающих различным начальным условиям, удовлетворяют дифференциальным уравнениям

$$\Delta\boldsymbol{r}'' - 2h\Delta\boldsymbol{r} = 0,\ \Delta r'' - 2h\Delta r = 0,\ \Delta t' = \Delta r.$$

Так как мы считаем h заданной и фиксированной и не зависящей от начальных условий, она общая для обоих решений. Введем обозначение $\omega = \sqrt{-2h}$, тогда

$$\Delta\boldsymbol{r}(s) = \Delta\boldsymbol{r}(0)\cos\omega s + \frac{\Delta\boldsymbol{r}'(0)}{\omega}\sin\omega s,$$

$$\Delta\boldsymbol{r}'(s) = -\Delta\boldsymbol{r}(0)\omega\sin\omega s + \Delta\boldsymbol{r}'(0)\cos\omega s,$$

такие же уравнения на $\Delta r(s)$ и $\Delta r'(s)$ и

$$\Delta t(s) = \Delta t(0) + \frac{\Delta r(0)}{\omega}\sin\omega s - \frac{\Delta r'(0)}{\omega^2}(1 - \cos\omega s).$$

Все эти приращения равномерно ограничены и могут быть сделаны сколь угодно малыми при уменьшении приращений в начальный момент времени.

Обратимся к результатам численных экспериментов. Исходные уравнения задачи двух тел и уравнения Шперлинга – Боде были проинтегрированы методами Дормана – Принса 5(4) и LSODA с начальными условиями $\boldsymbol{r}_0 = [1, 0, 0]$, $\boldsymbol{v}_0 = [0, 1, 0]$ на интервале времени $t \in [0, 10 \cdot 2\pi]$. Считается $\mu = 1$. Энергия и вектор Лапласа подавались на вход функции, реализующей функцию правых частей уравнений.

В таблицах приводятся нормы невязок по положению в конце интервала интегрирования по сравнению с начальными условиями в зависимости от абсолютной и относительной точности, а также число обращений в функциям правых частей. Из таблиц видно, что бывают ситуации, когда число обращений к функции правых частей уравнений Шперлинга – Боде меньше, чем к функции правых частей исходных уровнений, а точность в конце выше. В большинстве случаев точность получилась как минимум на два порядка лучше при интегрировании уравнений Шперлинга – Боде. Из таблиц также следует, что для достижения одной и той же точности методу LSODA требуется меньше обращений к функции правых частей, чем методу Дормана – Принса 5(4). Результаты в третьей строке первой таблицы объясняются попаданием решения в область, близкую к притягивающему центру вследствие ошибок метода интегрирования. Заметна также нерегулярность уменьшения невязки при повышении точности при интегрировании методом LSODA классических уравнений движения.

Таблица 3. Результаты численного интегрирования методом Дормана – Принса 5(4) десяти витков круговой орбиты с использованием классических уравнений и уравнений Шперлинга – Боде (SB)

$\lvert\Delta \boldsymbol{r}\rvert$	$\lvert\Delta \boldsymbol{r}_{SB}\rvert$	atol = rtol	nfev	nfev_{SB}
3.708486e+02	4.836508e+01	1.0E-01	32	182
6.515004e+01	9.482892e-01	1.0E-02	572	212
1.000000e+00	7.014320e-02	1.0E-03	7718	320
6.098573e-01	5.612642e-03	1.0E-04	560	500
3.054077e-01	4.770080e-04	1.0E-05	782	800
9.794763e-03	4.330681e-05	1.0E-06	1244	1274
1.481544e-04	4.091044e-06	1.0E-07	1994	2024
1.664302e-05	3.974176e-07	1.0E-08	3176	3218
2.901276e-06	3.912410e-08	1.0E-09	5048	5114
3.407411e-07	3.872976e-09	1.0E-10	8012	8108
3.623049e-08	3.850607e-10	1.0E-11	12710	12860
3.718764e-09	3.836880e-11	1.0E-12	20156	20396
3.766569e-10	3.829109e-12	1.0E-13	31952	32330

Таблица 4. Результаты численного интегрирования методом LSODA десяти витков круговой орбиты с использованием классических уравнений и уравнений Шперлинга – Боде (SB)

$\lvert\Delta \boldsymbol{r}\rvert$	$\lvert\Delta \boldsymbol{r}_{SB}\rvert$	atol = rtol	nfev	nfev_{SB}
1.411964e+05	1.339066e+00	1.0E-01	5911	242
1.590552e+00	7.051243e-01	1.0E-02	1190	293
2.144794e+00	1.081609e-01	1.0E-03	433	421
2.043476e+00	6.433252e-03	1.0E-04	613	453
5.519603e-02	5.482871e-04	1.0E-05	659	643
1.805570e-02	8.527356e-05	1.0E-06	847	653
3.556198e-03	4.892690e-06	1.0E-07	1199	803
4.005813e-05	1.123054e-06	1.0E-08	1193	949
7.183066e-06	1.648312e-07	1.0E-09	1485	1201
1.647786e-06	6.551653e-09	1.0E-10	1883	1231
4.275665e-07	2.839370e-09	1.0E-11	2135	1439
2.622624e-09	1.115154e-11	1.0E-12	2008	2034
1.371588e-10	1.707975e-12	1.0E-13	2847	2535

7.4. Уравнения Кустаанхеймо – Штифеля

Выполним преобразование Сундмана $dt = rds$ и будем считать s независимой переменной. Рассмотрим сначала плоское движение аппарата, то есть будем считать, что положение, скорость аппарата и возмущающее ускорение лежат все время в одной и той же плоскости. Пусть начало координат этой плоскости помещено в притягивающий центр, а координаты обозначаются как (x, y).

Идея вывода регуляризированных и устойчивых в задаче двух тел уравнений состоит в том, чтобы интерпретировать плоскость (x, y) как комплексную и рассмотреть *преобразование Леви-Чивиты*:

$$x + iy = (u_1 + iu_2)^2.$$

Вектор $\boldsymbol{r} = [x, y]^T$ связан с вектором $\boldsymbol{u} = [u_1, u_2]^T$ соотношением

$$\boldsymbol{r} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} u_1^2 - u_2^2 \\ 2u_1u_2 \end{bmatrix} = \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \boldsymbol{L}(\boldsymbol{u})\boldsymbol{u}.$$

Матрица $\boldsymbol{L}(\boldsymbol{u})$ называется матрицей *Леви-Чивиты*. Легко видеть, что

$$\boldsymbol{L}(\boldsymbol{u})' = \boldsymbol{L}(\boldsymbol{u}'),\ \boldsymbol{L}(\boldsymbol{u})^T\boldsymbol{L}(\boldsymbol{u}) = |\boldsymbol{u}|^2\boldsymbol{E}_2.$$

Кроме того, выполняются соотношения

$$\boldsymbol{L}(\boldsymbol{u})\boldsymbol{v} = \boldsymbol{L}(\boldsymbol{v})\boldsymbol{u}, \tag{16}$$

$$(\boldsymbol{u}^T\boldsymbol{u})\boldsymbol{L}(\boldsymbol{v})\boldsymbol{v} - 2(\boldsymbol{u}^T\boldsymbol{v})\boldsymbol{L}(\boldsymbol{u})\boldsymbol{v} + (\boldsymbol{v}^T\boldsymbol{v})\boldsymbol{L}(\boldsymbol{u})\boldsymbol{u} = 0 \tag{17}$$

для любых двумерных векторов $\boldsymbol{u}$ и $\boldsymbol{v}$. Из соотношения

$$r^2 = (u_1^2 - u_2)^2 + 4u_1^2u_2^2 = (u_1^2 + u_2^2)^2$$

получаем

$$r = u_1^2 + u_2^2 = |\boldsymbol{u}|^2.$$

Что касается производных,

$$\boldsymbol{r}' = (\boldsymbol{L}(\boldsymbol{u})\boldsymbol{u})' = \boldsymbol{L}(\boldsymbol{u}')\boldsymbol{u} + \boldsymbol{L}(\boldsymbol{u})\boldsymbol{u}' = 2\boldsymbol{L}(\boldsymbol{u})\boldsymbol{u}',\ \boldsymbol{u}' = \frac{1}{2\boldsymbol{u}^T\boldsymbol{u}}\boldsymbol{L}^T(\boldsymbol{u})\boldsymbol{r}'. \tag{18}$$

Из этих свойств матрицы $\boldsymbol{L}(\boldsymbol{u})$ и связи между $\boldsymbol{r}$ и $\boldsymbol{u}$ можно вывести уравнения движения в новых переменных[65]:

$$\boldsymbol{u}'' - \frac{h}{2}\boldsymbol{u} = \frac{|\boldsymbol{u}|^2}{2}\boldsymbol{L}^T(\boldsymbol{u})\boldsymbol{F},\ h' = 2\boldsymbol{u}' \cdot \boldsymbol{L}^T(\boldsymbol{u})\boldsymbol{F},\ t = |\boldsymbol{u}|^2, \tag{19}$$

[65] Штифель Е., Шейфеле Г. Линейная и регулярная небесная механика. Москва : Наука, 1975. С. 26–27.

где $h = (2|\boldsymbol{u}'|^2 - \mu)/|\boldsymbol{u}|^2 = v^2/2 - \mu/r$ – полная энергия.

Много сил и времени ушло у исследователей на обобщение преобразования, предложенного Леви-Чивитой, на пространственный случай движения аппарата. Попытки найти квадратную трехстрочную матрицу $\boldsymbol{L}$, обладающую схожими свойствами, оказались безуспешными, и была даже доказана теорема о том, что существование такой матрицы невозможно[66]. Кустаанхеймо[67] предложил использовать теорию спиноров, обобщений комплексных чисел. В этом случае, правда, приходится рассматривать четырехстрочную матрицу $\boldsymbol{L}(\boldsymbol{u})$:

$$\boldsymbol{L}(\boldsymbol{u}) = \begin{bmatrix} u_1 & -u_2 & -u_3 & u_4 \\ u_2 & u_1 & -u_4 & -u_3 \\ u_3 & u_4 & u_1 & u_2 \\ u_4 & -u_3 & u_2 & -u_1 \end{bmatrix}$$

и рассматривать преобразование

$$\boldsymbol{x} \doteq \begin{bmatrix} \boldsymbol{r} \\ 0 \end{bmatrix} = \boldsymbol{L}(\boldsymbol{u})\boldsymbol{u},$$

где $\boldsymbol{u} = [u_1, u_2, u_3, u_4]^T$. Для введенной таким образом матрицы $\boldsymbol{L}$ легко показать, что выполнены соотношения

$$r = |\boldsymbol{u}|^2, \;\; \boldsymbol{L}^T(\boldsymbol{u})\boldsymbol{L}(\boldsymbol{u}) = |\boldsymbol{u}|^2\boldsymbol{E}_4, \;\; \boldsymbol{L}'(\boldsymbol{u}) = \boldsymbol{L}(\boldsymbol{u}').$$

Соотношение $\boldsymbol{L}(\boldsymbol{u})\boldsymbol{v} = \boldsymbol{L}(\boldsymbol{v})\boldsymbol{u}$ может уже не выполняться, так как четвертые компоненты векторов справа и слева отличаются знаком. Но если эти компоненты равны нулю, то равенство выполняется. Итак, если два вектора $\boldsymbol{u}$ и $\boldsymbol{v}$ удовлетворяют так называемому *билинейному соотношению*

$$u_4v_1 - u_3v_2 + u_2v_3 - u_1v_4 = 0,$$

то $\boldsymbol{L}(\boldsymbol{u})\boldsymbol{v} = \boldsymbol{L}(\boldsymbol{v})\boldsymbol{u}$, то есть выполнено свойство (16). Можно показать, что из билинейного соотношения следует и свойство (17).

В случае плоского движения уравнения в новых переменных выводятся из уравнений в переменных $\boldsymbol{r}$, $\boldsymbol{r}'$. В пространственном случае это сделать не удается, так как нет взаимной однозначности между векторами $\boldsymbol{x}$ и $\boldsymbol{u}$. На самом деле каждому $\boldsymbol{x}$ соответствует целое однопараметрическое множество векторов $\boldsymbol{u}$, удовлетворяющих соотношению

[66] Штифель Е., Шейфеле Г. Линейная и регулярная небесная механика. Москва : Наука, 1975. Глава XI.

[67] Kustaanheimo P., Stiefel E. Perturbation theory of Kepler Motion based on spinor regularization // J. Reine Angew. Math. 1965. Vol. 218 P. 204–219.

$\boldsymbol{x} = \boldsymbol{L}(\boldsymbol{u})\boldsymbol{u}$. Поэтому предлагается другая идея: постулируем уравнения (19) и докажем, что функция $\boldsymbol{r}(s)$, соответствующая решению $\boldsymbol{u}(s)$, удовлетворяет исходной системе уравнений.

Выберем $\boldsymbol{u}(0)$ произвольно по $\boldsymbol{x}(0)$, лишь бы было выполнено $\boldsymbol{x}(0) = \boldsymbol{L}(\boldsymbol{u}(0))\boldsymbol{u}(0)$. Например, компоненты $\boldsymbol{u}(0)$ можно вычислить по формулам[68]

$$u_1 = \sqrt{\frac{1}{2}(r + x_1)},\ u_2 = \frac{x_2 u_1}{r + x_1},\ u_3 = \frac{x_3 u_1}{r + x_1},\ u_4 = 0,$$

если $x_1 \geq 0$, и по формулам

$$u_2 = \sqrt{\frac{1}{2}(r - x_1)},\ u_3 = 0,\ u_1 = \frac{x_2 u_2}{r - x_1},\ u_4 = \frac{x_3 u_2}{r - x_1},$$

иначе. Скорость $\boldsymbol{u}'(0)$ вычислим по формуле (18), как в случае плоского движения:

$$\boldsymbol{u}'(0) = \frac{1}{2|\boldsymbol{u}(0)|^2}\boldsymbol{L}^T(\boldsymbol{u}(0))\boldsymbol{r}'(0).$$

Теперь заметим, что для произвольного вектора $\boldsymbol{y}$ с нулевой четвертой компонентой векторы $\boldsymbol{v} = \boldsymbol{L}^T(\boldsymbol{u})\boldsymbol{y}$ и $\boldsymbol{u}$ удовлетворяют билинейному соотношению. Действительно, векторы $\boldsymbol{v}$ и $\boldsymbol{u}$ удовлетворяют билинейному соотношению, если четвертая компонента вектора $\boldsymbol{L}(\boldsymbol{u})\boldsymbol{v}$ или $\boldsymbol{L}(\boldsymbol{v})\boldsymbol{u}$ (они отличаются только знаком) равна нулю, но

$$\boldsymbol{L}(\boldsymbol{u})\boldsymbol{v} = \boldsymbol{L}(\boldsymbol{u})\boldsymbol{L}^T(\boldsymbol{u})\boldsymbol{y} = |\boldsymbol{u}|^2\boldsymbol{y}$$

и имеет нулевую четвертую компоненту. Отсюда сразу следует, что $\boldsymbol{u}(0)$ и $\boldsymbol{u}'(0)$ по построению удовлетворяют билинейному соотношению. Более того, правая часть первого уравнения (19) удовлетворяет билинейному соотношению с вектором $\boldsymbol{u}$. Введем функцию $l(\boldsymbol{u}, \boldsymbol{u}') = u_4 u_1' - u_3 u_2' + {} + u_2 u_3' - u_1 u_4'$ и продифференцируем ее по времени. Тогда окажется

$$\frac{d}{ds}l = u_4 u_1'' - u_3 u_2'' + u_2 u_3'' - u_1 u_4'' = 0$$

после подстановки туда выражений для вторых производных, взятых из уравнения (19). Это значит, что $l(\boldsymbol{u}, \boldsymbol{u}')$ является первым интегралом уравнений (19) и сохраняет значения на решениях этих уравнений. Но в начальный момент времени $l(\boldsymbol{u}(0), \boldsymbol{u}'(0)) = 0$, так как начальные условия удовлетворяют билинейному соотношению. Значит, и во всякий

[68] Штифель Е., Шейфеле Г. Линейная и регулярная небесная механика. Москва : Наука, 1975. С. 38.

другой момент времени будет $l(\boldsymbol{u}(s), \boldsymbol{u}'(s)) = 0$, а это значит, что в любой момент s векторы $\boldsymbol{u}(s)$ и $\boldsymbol{u}'(s)$ удовлетворяют билинейному соотношению, что влечет

$$\boldsymbol{L}(\boldsymbol{u})\boldsymbol{u}' = \boldsymbol{L}(\boldsymbol{u}')\boldsymbol{u}.$$

Отсюда следует важный результат:

$$\boldsymbol{x}' = 2\boldsymbol{L}(\boldsymbol{u})\boldsymbol{u}'.$$

Действительно, равенство первых трех компонент этого равенства следует из простейших свойств матрицы $\boldsymbol{L}$. Равенство же четвертых компонент следует из билинейного соотношения между $\boldsymbol{u}$ и $\boldsymbol{u}'$. Наконец, сформулирую теорему о связи решений в новых и исходных переменных[69].

Теорема. *Пусть $\boldsymbol{u}(s)$ – решение системы уравнений*

$$\boldsymbol{u}'' - \frac{h}{2}\boldsymbol{u} = \frac{|\boldsymbol{u}|^2}{2}\boldsymbol{L}^T(\boldsymbol{u})\boldsymbol{F},\ h' = 2\boldsymbol{u}' \cdot \boldsymbol{L}^T(\boldsymbol{u})\boldsymbol{F},\ t = |\boldsymbol{u}|^2,$$

с начальными условиями $\boldsymbol{u}(0)$ и $\boldsymbol{u}'(0)$, отвечающими начальным условиям $\boldsymbol{x}(0) = [\boldsymbol{r}(0), 0]$ и $\boldsymbol{x}'(0) = [\boldsymbol{r}'(0), 0]$ так, как описано выше. Тогда функция $\boldsymbol{r}(s)$, как первые три компоненты вектора

$$\boldsymbol{x}(s) = \boldsymbol{L}(\boldsymbol{u}(s))\boldsymbol{u}(s),$$

является решением исходной системы уравнений

$$\boldsymbol{r}' = r\boldsymbol{v},\ \boldsymbol{v}' = -\mu\boldsymbol{r}/r^2 + r\boldsymbol{F},\ t' = r$$

с начальными условиями $\boldsymbol{r}(0)$ и $\boldsymbol{v}(0) = \boldsymbol{r}'(0)/r(0)$.

Уравнения выше можно дополнить уравнением на $r = |\boldsymbol{u}|^2$, чтобы сделать их линейными в невозмущенном случае:

$$r'' - 2hr - \mu = r\boldsymbol{F}^T\boldsymbol{L}(\boldsymbol{u})\boldsymbol{u} = r\boldsymbol{u}^T\boldsymbol{L}^T(\boldsymbol{u})\boldsymbol{F}.$$

Итак, полная система уравнений выглядит так:

$$\boldsymbol{u}'' - \frac{h}{2}\boldsymbol{u} = \frac{|\boldsymbol{u}|^2}{2}\boldsymbol{L}^T(\boldsymbol{u})\boldsymbol{F},\ h' = 2\boldsymbol{u}' \cdot \boldsymbol{L}^T(\boldsymbol{u})\boldsymbol{F},\ t' = r,$$

$$r'' - 2hr - \mu = r\boldsymbol{u}^T\boldsymbol{L}^T(\boldsymbol{u})\boldsymbol{F}.$$

[69] Штифель Е., Шейфеле Г. Линейная и регулярная небесная механика. Москва : Наука, 1975. С. 31.

Ее можно переписать как систему первого порядка:

$$\boldsymbol{u}' = \boldsymbol{w},\ \boldsymbol{w}' = \frac{h}{2}\boldsymbol{u} + \frac{|\boldsymbol{u}|^2}{2}\boldsymbol{L}^T(\boldsymbol{u})\boldsymbol{F},$$

$$r' = q,\ q' = 2hr + \mu + r\boldsymbol{u}^T\boldsymbol{L}^T(\boldsymbol{u})\boldsymbol{F},\ h' = 2\boldsymbol{u}'\cdot\boldsymbol{L}^T(\boldsymbol{u})\boldsymbol{F},\ t' = r.$$

Эти уравнения называются *уравнениями Кустаанхеймо – Штифеля*, а переменные $\boldsymbol{u}$ называют *переменными Кустаанхеймо – Штифеля* (*KS-переменными*).

По аналогии с уравнениями Шперлинга – Боде легко видеть, что в невозмущенном случае и при исключении уравнения для энергии h эллиптическое движение устойчиво. В таблицах ниже приводятся результаты численного моделирования круговой орбиты в исходных уравнениях и в уравнениях Кустаанхеймо – Штифеля. Условия моделирования те же, что в предыдущем разделе. Видно, что во всех случаях точность интегрирования уравнений Кустаанхеймо – Штифеля выше, причем, как правило, на два порядка, а число обращений к функции правой части в два раза меньше. Число обращений к функции правой части будет меньше, если сравнивать эти результаты с результатами интегрирования уравнений Шперлинга – Боде. Это можно объяснить тем, что в уравнениях Кустаанхеймо – Штифеля частота переменной $\boldsymbol{u}$ в два раза меньше, что делает гармонические зависимости более пологими. Частота изменения функции $r = r(s)$ имеет исходную частоту, но в случае интегрирования круговой орбиты $r' \approx 0$, так что уравнение для r точность интегрирования не ухудшает. Если же интегрировать высокоэллиптическую орбиту с эксцентриситетом $e = 0.8$, то можно увидеть, что достижение той же точности, что и для круговой орбиты, требует примерно в два раза больше обращений к функции правой части и результаты становятся похожи на результаты при интегрировании уравнений Шперлинга – Боде. Замечу также, что в случае эллиптической орбиты если $r \in [r_{\min}, r_{\max}]$, то $u_i \in [\sqrt{r_{\min}}, \sqrt{r_{\max}}]$ и

$$\sqrt{r_{\max}} - \sqrt{r_{\min}} = \frac{r_{\max} - r_{\min}}{\sqrt{r_{\max}} + \sqrt{r_{\min}}} \le r_{\max} - r_{\min},$$

если $\sqrt{r_{\max}} + \sqrt{r_{\min}} \ge 1$, например, при $r_{\min} \ge 1$. Это значит, что в KS-переменных в этих случаях амплитуда зависимых переменных u_i уменьшается.

7.5. Уравнения в орбитальных элементах

В задаче двух тел известно множество интегралов движения – функций положения и скорости, которые являются постоянными на

Таблица 5. Результаты численного интегрирования методом Дормана – Принса 5(4) десяти витков круговой орбиты с использованием классических уравнений и уравнений Кустаанхеймо – Штифеля (KS)

$\lvert\Delta \boldsymbol{r}\rvert$	$\lvert\Delta \boldsymbol{r}_{KS}\rvert$	atol = rtol	nfev	nfev_{KS}
3.708486e+02	3.174569e+02	1.0E-01	32	80
6.515004e+01	1.525253e+00	1.0E-02	572	110
1.000000e+00	9.159165e-02	1.0E-03	7718	158
6.098573e-01	7.469204e-03	1.0E-04	560	248
3.054077e-01	6.386144e-04	1.0E-05	782	386
9.794763e-03	5.759005e-05	1.0E-06	1244	608
1.481544e-04	5.477892e-06	1.0E-07	1994	962
1.664302e-05	5.354930e-07	1.0E-08	3176	1526
2.901276e-06	5.255792e-08	1.0E-09	5048	2414
3.407411e-07	5.216424e-09	1.0E-10	8012	3824
3.623049e-08	5.185395e-10	1.0E-11	12710	6062
3.718764e-09	5.169663e-11	1.0E-12	20156	9608
3.766569e-10	5.165945e-12	1.0E-13	31952	15230

Таблица 6. Результаты численного интегрирования методом LSODA десяти витков круговой орбиты с использованием классических уравнений и уравнений Кустаанхеймо – Штифеля (KS)

$\lvert\Delta \boldsymbol{r}\rvert$	$\lvert\Delta \boldsymbol{r}_{KS}\rvert$	atol = rtol	nfev	nfev_{KS}
1.411964e+05	1.199882e+00	1.0E-01	5911	125
1.590552e+00	7.005444e-01	1.0E-02	1190	146
2.144794e+00	1.121410e-01	1.0E-03	433	213
2.043476e+00	6.218669e-03	1.0E-04	613	229
5.519603e-02	5.334332e-04	1.0E-05	659	331
1.805570e-02	9.040609e-05	1.0E-06	847	341
3.556198e-03	5.082899e-06	1.0E-07	1199	421
4.005813e-05	1.082693e-06	1.0E-08	1193	509
7.183066e-06	1.626318e-07	1.0E-09	1485	625
1.647786e-06	6.501466e-09	1.0E-10	1883	639
4.275665e-07	2.301747e-09	1.0E-11	2135	759
2.622624e-09	1.856870e-11	1.0E-12	2008	1008
1.371588e-10	1.833466e-12	1.0E-13	2847	1295

решениях системы уравнений. К ним относятся интеграл энергии h, интеграл площадей $\boldsymbol{c}$, интеграл Лапласа $\boldsymbol{A}$, а также так называемые *орбитальные элементы* – долгота восходящего узла Ω, наклонение орбиты i, аргумент перицентра ω, эксцентриситет e, фокальный параметр p, большая полуось a, момент времени прохождения через перицентр τ_π, средняя аномалия в начальный момент времени M_0 и произвольные непрерывно дифференцируемые функции орбитальных элементов. Все эти интегралы движения являются функциями положения и скорости аппарата и сохраняются на решениях задачи двух тел. На решениях возмущенной задачи двух тел эти функции могут не сохранять свои значения и являются, вообще говоря, функциями времени. Мы уже выводили эволюционные уравнения, которым подчиняются интеграл энергии, интеграл площадей и интеграл Лапласа. Похожим образом можно вывести и уравнения для орбитальных элементов и функций от них, для этого нужно только выразить интересующий элемент орбиты в терминах положения и скорости, продифференцировать по времени и учесть возмущенную систему уравнений для положения и скорости. В теории дифференциальных уравнений известен общий метод, связывающий решения возмущенных и невозмущенных уравнений движения – *метод Лагранжа вариации постоянных*, напомню его.

Пусть дана система дифференциальных уравнений

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}),\ \boldsymbol{x} \in \mathbb{R}^n$$

с известным решением $\boldsymbol{x}_0(t) = \boldsymbol{x}_0(t; C_1, \ldots, C_n)$, где C_1, …, C_n – постоянные интегрирования или любые постоянные, однозначно определяющие решение. Тогда решение возмущенной системы уравнений

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}) + \boldsymbol{g}(t, \boldsymbol{x}),$$

можно искать в виде $\boldsymbol{x}(t) = \boldsymbol{x}_0(t; C_1(t), \ldots, C_n(t))$, то есть в виде решения невозмущенной системы уравнений с интерпретацией C_1, …, C_n как функций времени, поведение которых, вообще говоря, зависит от функции $\boldsymbol{g}$. Определим уравнения, которым подчиняются эти функции. Для этого введем вектор-функцию $\boldsymbol{C}(t) = [C_1(t), \ldots, C_n(t)]$ и подставим функцию $\boldsymbol{x}(t) = \boldsymbol{x}_0(t; \boldsymbol{C}(t))$ в возмущенные уравнения:

$$\frac{\partial \boldsymbol{x}_0}{\partial t} + \frac{\partial \boldsymbol{x}_0}{\partial \boldsymbol{C}} \dot{\boldsymbol{C}} = \boldsymbol{f}(t, \boldsymbol{x}_0(t; \boldsymbol{C}(t))) + \boldsymbol{g}(t, \boldsymbol{x}_0(t; \boldsymbol{C}(t))).$$

Заметим, что первые слагаемые в левой и правой частях уравнения равны в любой момент времени. Предположим, что матрица $\partial \boldsymbol{x}_0 / \partial \boldsymbol{C}$

обратима, тогда

$$\dot{\boldsymbol{C}} = \left(\frac{\partial \boldsymbol{x}_0}{\partial \boldsymbol{C}}\right)^{-1} \boldsymbol{g}(t, \boldsymbol{x}_0(t; \boldsymbol{C}(t))).$$

Это и есть искомые эволюционные уравнения для функций $C_1(t)$, …, $C_n(t)$.

С точки зрения задачи двух тел постоянными C_1, …, C_n могут быть орбитальные элементы, интеграл энергии, интеграл площадей, интеграл Лапласа и функции от них. В принципе, постоянными C_1, …, C_n могут быть начальные положение $\boldsymbol{r}_0$ и скорость аппарата $\boldsymbol{v}_0$, так как движение однозначно определяется начальными условиями.

Итак, мы смотрим на решения возмущенных уравнений движения как на решения невозмущенных уравнений, но с меняющимися параметрами, определяющими невозмущенное движение. Под действием возмущений от времени будут зависеть большая полуось, эксцентриситет орбиты, наклонение орбиты, полная энергия, вектор Лапласа и другие функции, которые являются интегралами движения в невозмущенной задаче двух тел. Иногда такие функции времени называют *оскулирующими* («касающимися» в переводе с латинского языка), потому что в каждый момент времени они определяют орбиту невозмущенных уравнений движения, по которой двигался бы аппарат, если бы начиная с этого момента возмущение исчезло. Такой взгляд на вещи очень удобен для анализа возмущенной траектории, ведь *форма* зависимости решений от C_i сохраняется, просто C_i зависят от времени, причем зависят от времени слабо, если функция $\boldsymbol{g}$ по модулю принимает малые значения. Получается, что метод вариации постоянных позволяет нам *сохранить привычный вид* решений из невозмущенной задачи двух тел.

Уравнения на орбитальные элементы обычно выводятся в курсе механики космического полета. Мы же рассмотрим их здесь без вывода, но подробно изучим их свойства. Итак, уравнения движения в *классических орбитальных элементах* a, e, i, Ω, ω, M_0 имеют следующий вид[70]:

$$\frac{da}{dt} = \frac{2a^2}{\sqrt{\mu p}}\left(e\sin\vartheta \cdot S + \frac{p}{r}\cdot T\right),$$

$$\frac{de}{dt} = \sqrt{\frac{p}{\mu}}\left\{\sin\vartheta \cdot S + \left[\left(1+\frac{r}{p}\right)\cos\vartheta + e\frac{r}{p}\right]\cdot T\right\},$$

[70]Субботин М.Ф. Введение в теоретическую астрономию. Москва : Наука, 1968. С. 507.

$$\frac{di}{dt} = \frac{r\cos(\omega+\vartheta)}{\sqrt{\mu p}} \cdot W,$$

$$\frac{d\Omega}{dt} = \frac{r\sin(\omega+\vartheta)}{\sqrt{\mu p}\sin i} \cdot W,$$

$$\frac{d\omega}{dt} = \sqrt{\frac{p}{\mu}}\left[-\frac{\cos\vartheta}{e}\cdot S + \left(1+\frac{r}{p}\right)\frac{\sin\vartheta}{e}\cdot T - \frac{r}{p}\sin(\omega+\vartheta)\operatorname{ctg} i\cdot W\right],$$

$$\frac{dM_0}{dt} = \sqrt{\frac{p}{\mu}}\frac{\sqrt{1-e^2}}{e}\left[\left(\cos\vartheta - 2e\frac{r}{p}\right)\cdot S - \left(1+\frac{r}{p}\right)\sin\vartheta\cdot T\right],$$

где $S = F_r$, $T = F_t$, $W = F_c$ – компоненты возмущающего ускорения $\boldsymbol{F}$ по радиус-вектору $\boldsymbol{r}$, трансверсали $\boldsymbol{c}\times\boldsymbol{r} = (\boldsymbol{r}\times\boldsymbol{v})\times\boldsymbol{r}$ и нормали к плоскости орбиты $\boldsymbol{c} = \boldsymbol{r}\times\boldsymbol{v}$. Систему координат, образованную этими тремя направлениями, еще называют *орбитальной системой координат*. Угол M_0 в этих уравнениях – это угол из тождества

$$M(t) = M_0 + \int_{t_0}^{t} n\,dt,\ n = \sqrt{\frac{\mu}{a^3}},$$

он совпадает с начальным значением средней аномалии $M(t_0)$ в невозмущенной задаче двух тел, а в возмущенной задаче двух тел является функцией времени. Всюду в уравнениях выше $r = p/(1+e\cos\vartheta)$, а $\vartheta = \vartheta(t)$ – это угол истинной аномалии. Этот угол можно найти из угла эксцентрической аномалии $E(t)$, который, в свою очередь, находится из решения уравнения Кеплера:

$$E - e\sin E = M,$$

которое решается при каждом обращении к функции правой части уравнений. Эти проблемы можно обойти, если вместо уравнения на M_0 рассматривать сразу уравнение на M:

$$\frac{dM}{dt} = \sqrt{\frac{\mu}{a^3}} + \sqrt{\frac{p}{\mu}}\frac{\sqrt{1-e^2}}{e}\left[\left(\cos\vartheta - 2e\frac{r}{p}\right)\cdot S - \left(1+\frac{r}{p}\right)\sin\vartheta\cdot T\right],$$

но решать уравнение Кеплера, чтобы найти эксцентрическую аномалию, а по ней и истинную аномалию, все равно придется. Эту проблему можно обойти заменой уравнения для средней аномалии на уравнение для истинной аномалии:

$$\frac{d\vartheta}{dt} = \frac{\sqrt{\mu p}}{r^2} + \sqrt{\frac{p}{\mu}}\left[\frac{\cos\vartheta}{e}\cdot S - \left(1+\frac{r}{p}\right)\frac{\sin\vartheta}{e}\cdot T\right].$$

Уравнения выше на орбитальные уравнения называются *уравнениями Эйлера*, который впервые вывел (почти все) их и применил для анализа движения Луны в середине XVIII века[71]. Небольшое время спустя Лагранж завершил вывод оставшихся уравнений и рассмотрел частный и полезный для астрономических приложений случай – когда возмущающее ускорение представляет собой градиент некоторой скалярной функции (пертурбационной функции). Эти уравнения называются *уравнениями Лагранжа*, они имеют свои интересные особенности[72]. Наконец, в XIX веке Гаусс использовал уравнения Эйлера для исследования движения Паллады, астероида из главного пояса астероидов. Энке изложил метод в Приложениях к Берлинскому астрономическому ежегоднику на 1837 и 1838 гг., и с тех пор, возможно, указанные уравнения Эйлера часто стали называться *уравнениями Гаусса*.

Уравнения в орбитальных элементах удобны для анализа эволюции орбиты под действием возмущений. Из их вида сразу понятно, как те или иные возмущения влияют на орбитальные элементы. Например, из уравнений видно, что радиальная и трансверсальная компоненты возмущения не оказывают влияния на наклонение орбиты и долготу восходящего узла. Видно, что нормальная компонента возмущения не оказывает влияния на большую полуось и эксцентриситет. Уравнения можно использовать для оценок изменения орбитальных элементов со временем без их непосредственного интегрирования, что обычно возможно только с использованием численных методов. При отсутствии возмущений уравнения имеют максимально простой вид – правые части всех уравнений (кроме только что уравнения для M) равны нулю, а правая часть уравнения для M равна константе n. Поэтому если n зафиксировать, то решения этих уравнений являются устойчивыми по Ляпунову и, в силу равенства нулю правых частей, интегрируются максимально быстро по сравнению с любыми другими уравнениями. Если возмущающее ускорение имеет небольшую величину и пологий график функции времени, то такими будут и правые части уравнений. Усложнить задачу методу интегрирования способно уравнение для истинной аномалии, так как оно не пропорционально возмущающему ускорению и содержит r в знаменателе, который для высокоэллиптических орбит будет меняться значительно. Замечу также, что если возмущающее ускорение имеет достаточно высокие частоту и амплитуду, то

[71] Субботин М.Ф. Введение в теоретическую астрономию. Москва : Наука, 1968. С. 535–537

[72] Субботин М.Ф. Введение в теоретическую астрономию. Москва : Наука, 1968. С. 510–514

каких-либо значительных преимуществ интегрирования этой системы уравнений перед интегрированием уравнений движения в терминах $\boldsymbol{r}$ и $\boldsymbol{v}$ может не быть.

Приведенные уравнения орбитального движения имеют и существенные недостатки – они вырождаются при $e = 0$ и $\sin i = 0$, то есть для круговых и экваториальных орбит. Дело в том, что для таких орбит сами орбитальные элементы определены не в полном составе. Действительно, у круговых орбит не определен аргумент перицентра ω, а у экваториальных орбит не определена долгота восходящего узла Ω. Более того, указанные уравнения имеют смысл только для эллиптического движения. Существуют, впрочем, другие, невырождающиеся орбитальные элементы и пригодные для всех классов орбит.

Модифицированными равноденственными элементами[73] (*modified equinoctial elements*) называются

$$
\begin{aligned}
p &= a\left(1 - e^2\right), \\
e_x &= e\cos(\omega + \Omega), \\
e_y &= e\sin(\omega + \Omega), \\
i_x &= \operatorname{tg}(i/2)\cos\Omega, \\
i_y &= \operatorname{tg}(i/2)\sin\Omega, \\
L &= \vartheta + \omega + \Omega.
\end{aligned}
$$

Здесь L – истинная долгота. Считается, что эти зависимости по непрерывности продолжаются при $e \to 0$ и $i \to 0$, когда ω и Ω перестают быть определенными. Эти зависимости вырождаются только при $i = \pi$, но это легко исправить, немного скорректировав зависимости[74].

Равноденственные элементы можно связать непосредственно с положением и скоростью аппарата[75]. Пусть дан набор равноденственных

[73] Walker M.J.H., Ireland B. Owens, J. A set modified equinoctial orbit elements // Celestial mechanics. 1985. Vol. 36, N. 4. P. 409–419.

[74] Вводится так называемый *ретроградный фактор*, подробнее см. в монографии Иванов Д.С., Трофимов С.П., Широбоков М.Г. Численное моделирование орбитального и углового движения космических аппаратов. Москва : ИПМ им. М.В. Келдыша РАН, 2016. С. 104.

[75] Cefola P. Equinoctial orbit elements-Application to artificial satellite orbits // Astrodynamics Conference. September 1972. P. 937.

элементов p, e_x, e_y, i_x, i_y, L. Введем вспомогательные векторы

$$\boldsymbol{f} = \frac{\sqrt{\mu p}}{1+i_x^2+i_y^2}\begin{bmatrix} 1+i_x^2-i_y^2 \\ 2i_xi_y \\ -2i_y \end{bmatrix},\ \boldsymbol{g} = \frac{\sqrt{\mu p}}{1+i_x^2+i_y^2}\begin{bmatrix} 2i_xi_y \\ 1-i_x^2+i_y^2 \\ 2i_x \end{bmatrix}.$$

Тогда положение и скорость аппарата определяются по формулам

$$\boldsymbol{r} = \frac{\sqrt{p/\mu}(\boldsymbol{f}\cos L + \boldsymbol{g}\sin L)}{1+e_x\cos L + e_y\sin L},\ \boldsymbol{v} = \frac{e_x+\cos L}{p}\boldsymbol{g} - \frac{e_y+\sin L}{p}\boldsymbol{f}.$$

Наоборот, пусть даны положение $\boldsymbol{r}$ и скорость $\boldsymbol{v}$ аппарата. По ним определим векторы Лапласа $\boldsymbol{A}$, орбитальный момент $\boldsymbol{c}$ и векторы

$$\boldsymbol{f} = \begin{bmatrix} c - \dfrac{c_1^2}{c+c_3} \\ -\dfrac{c_1c_2}{c+c_3} \\ -c_1 \end{bmatrix},\ \boldsymbol{g} = \begin{bmatrix} -\dfrac{c_1c_2}{c+c_3} \\ c - \dfrac{c_2^2}{c+c_3} \\ -c_2 \end{bmatrix},$$

где $\boldsymbol{c} = \boldsymbol{r}\times\boldsymbol{v} = [c_1,c_2,c_3]^T$, $c = |\boldsymbol{c}|$. Тогда

$$p = \frac{c^2}{\mu},\ e_x = \frac{\boldsymbol{A}\cdot\boldsymbol{f}}{c\mu},\ e_y = \frac{\boldsymbol{A}\cdot\boldsymbol{g}}{c\mu},\ i_x = -\frac{c_2}{c+c_3},\ i_y = \frac{c_1}{c+c_3},$$

$$L = \operatorname{atan2}(\boldsymbol{r}\cdot\boldsymbol{g}, \boldsymbol{r}\cdot\boldsymbol{f}),$$

где $\operatorname{atan2}(y,x)$ – используемая во многих программных математических пакетах функция, сопоставляющая точке (x,y) на координатной плоскости угол между направлением на эту точку и осью x.

Легко также получить выражения для классических орбитальных элементов в терминах равноденственных элементов:

$$e = \sqrt{e_x^2+e_y^2},\ i = 2\operatorname{arctg}\sqrt{i_x^2+i_y^2},\ \Omega = \operatorname{atan2}(i_y,i_x),$$

$$\omega = \operatorname{atan2}(e_y,e_x) - \Omega,\ \vartheta = L - \omega - \Omega.$$

Уравнения в равноденственных элементах имеют следующий вид:

$$\frac{dp}{dt} = \frac{2p}{\sigma}\sqrt{\frac{p}{\mu}} \cdot T,$$

$$\frac{de_x}{dt} = \sqrt{\frac{p}{\mu}}\left\{\sin L \cdot S + \left[\left(1 + \frac{1}{\sigma}\right)\cos L + \frac{e_x}{\sigma}\right] \cdot T - \frac{e_y \eta}{\sigma} \cdot W\right\},$$

$$\frac{de_y}{dt} = \sqrt{\frac{p}{\mu}}\left\{-\cos L \cdot S + \left[\left(1 + \frac{1}{\sigma}\right)\sin L + \frac{e_y}{\sigma}\right] \cdot T + \frac{e_x \eta}{\sigma} \cdot W\right\},$$

$$\frac{di_x}{dt} = \sqrt{\frac{p}{\mu}}\frac{\varphi \cos L}{2\sigma} \cdot W,$$

$$\frac{di_y}{dt} = \sqrt{\frac{p}{\mu}}\frac{\varphi \sin L}{2\sigma} \cdot W,$$

$$\frac{dL}{dt} = \sqrt{\frac{\mu}{p}}\frac{\sigma^2}{p} + \sqrt{\frac{p}{\mu}}\frac{\eta}{\sigma} \cdot W,$$

где $\sigma = 1 + e_x \cos L + e_y \sin L$, $\eta = i_x \sin L - i_y \cos L$, $\varphi = 1 + i_x^2 + i_y^2$. Эти уравнения не вырождаются при $e = 0$ и $\sin i = 0$ и не требуют расчета истинной аномалии и решения уравнения Кеплера. Впрочем, эти уравнения все же не лишены недостатков. Из-за наличия функции σ истинная долгота на эллиптической орбите меняется неравномерно, что вынуждает метод интегрирования варьировать шаг интегрирования.

Функции преобразования из декартовых координат и скоростей в классические и равноденственные элементы орбиты и обратно реализованы автором пособия на Fortran в библиотеке KIAM Astrodynamics Toolbox[76]. Эти функции имеют названия `kiam.rv2oe`, `kiam.oe2rv` для орбитальных элементов и `kiam.rv2ee`, `kiam.ee2rv` для равноденственных элементов.

8. Вопросы на понимание

1. Что делать в случае недоопределенных или переопределенных систем уравнений?
2. Каковые условия сходимости метода градиентного спуска? Метода Ньютона? Метода простых итераций?
3. Как связаны метод Ньютона и метод простых итераций?

[76] https://github.com/shmaxg/KIAMToolbox.

4. В методе простых итераций $\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k)$ производная $\partial \boldsymbol{f}/\partial \boldsymbol{x}$ по одной норме оказалась больше единицы, а по другой норме меньше единицы. Будет ли отображение $\boldsymbol{f}$ сжимающим?
5. Каковы предельные случаи метода Левенберга – Марквардта при $\lambda \to 0$ и $\lambda \to \infty$?
6. В каких случаях уместно применять градиентные, а в каких безградиентные методы оптимизации?
7. Каким образом ограничения-неравенства можно переписать в виде ограничений-равенств? А наоборот?
8. Что общего и чем отличаются метод линейного поиска и метод доверительных областей для выбора шага метода оптимизации?
9. Сколько в среднем нужно сделать итераций в методе чистого случайного поиска, чтобы попасть в окрестность локального минимума, занимающую 5% объема от всей области поиска?
10. Как адаптивно менять коэффициенты метода роя частиц для повышения степени исследуемости пространства значений переменных и для повышения точности решения?
11. Что известно про сходимость метода Нелдера – Мида в задаче оптимизации с одной переменной, с двумя переменными?
12. В классе каких функций метод байесовской оптимизации аппроксимирует целевую функцию?
13. В чем отличие метода байесовской оптимизации от классического метода линейной регрессии?
14. В чем состоит проблема исследования и использования в методе байесовской оптимизации и как она решается?
15. В каких областях фазового пространства справедливы условия теорм существования и единственности решений уравнений движения космического аппарата?
16. Как соотносится локальная и глобальная погрешности методов интегрирования Рунге – Кутты, Адамса?
17. Как происходит управление шагом интегрирования во вложенных методах Рунге – Кутты?
18. Как аккуратно рассчитать траекторию движения аппарата, когда он периодически пересекает терминатор тени Земли?
19. Как в линейном приближении изменится положение аппарата в конечный момент времени при малом изменении его скорости в начальный момент времени? Описать методику расчета изменения положения.
20. Как проинтегрировать уравнения Шперлинга – Боде и Кустаанхеймо – Штифеля на заданном интервале физического времени?
21. В чем преимущество уравнений Шперлинга – Боде и Кустаан-

хеймо – Штифеля перед классическими уравнениями движения аппарата?

22. В чем преимущество уравнений Кустаанхеймо – Штифеля перед уравнениями Шперлинга – Боде?
23. Зачем нужно билинейное соотношение при выводе уравнений Кустаанхеймо – Штифеля?
24. В чем причина вырождений классических орбитальных элементов? Какие орбитальные элементы не вырождаются?

9. Задачи и упражнения

9.1. Числа, векторы и матрицы

Задача 1. Вычислить значение выражения

$$\frac{\operatorname{tg} 3 + \sqrt{e^{2.5} + \cos 4.1\pi}}{(\sin 2.6\pi - 3 \operatorname{tg} 3)^3} - \arcsin 0.4 \cdot \ln(2(1 + \cos 1.1\pi)).$$

Задача 2. Вычислить значение выражения

$$\frac{\dfrac{\sqrt{\ln 3}}{\cos 5.1\pi} + \sqrt{\dfrac{\operatorname{ch} 1.2\pi}{\ln 2.4}}}{\sqrt{\dfrac{\ln 6\pi}{\ln 2.5}}} \left| -10e^{-\sqrt{2}} + \sin 80^\circ \operatorname{tg} 27^\circ \right|.$$

Задача 3. Вычислить значение выражения

$$\left| e^{\sin 3.1\pi} - \pi^{e/2} \right| \cdot \arccos 0.1 \cdot \operatorname{th} 1 - \cos 3^\circ \cdot \ln(e^\pi - \pi^2).$$

Задача 4.

1. Вычислить сумму векторов $\boldsymbol{v}_1 = [2.1, 6.7, 3.1]$ и $\boldsymbol{v}_2 = [5.9, -4.7, 1.1]$.
2. Вывести третий элемент суммы $\boldsymbol{v}_1 + \boldsymbol{v}_2$.
3. Вычислить скалярное произведение $\boldsymbol{v}_1 \cdot \boldsymbol{v}_2$.
4. Записать в новый массив $\boldsymbol{w}$ первую и вторую компоненты $\boldsymbol{v}_1 - \boldsymbol{v}_2$.

Задача 5.

1. Завести вектор $\boldsymbol{v} = [5, 9, -1, 2, 8, 2, 0, 4]$.

2. Завести новый вектор $\boldsymbol{w} = \boldsymbol{v}$.
3. Заменить с 3-й по 5-ю компоненты $\boldsymbol{w}$ нулями.
4. Вычислить сумму всех элементов $\boldsymbol{w}$.
5. Увеличить все элементы $\boldsymbol{w}$ на 3.
6. Соединить последовательно векторы $\boldsymbol{v}$ и $\boldsymbol{w}$, получить вектор $\boldsymbol{e}$.
7. Найти максимальную и минимальную компоненты вектора $\boldsymbol{e}$.
8. Отсортировать по возрастанию массив $\boldsymbol{e}$.

Задача 6.

1. Завести вектор $\boldsymbol{v}$ чисел от 1 до 100.
2. Возвести все компоненты $\boldsymbol{v}$ в квадрат, получить вектор $\boldsymbol{w}$.
3. Заменить компоненты $\boldsymbol{w}$ с нечетными номерами на нули.
4. К компонентам $\boldsymbol{w}$ с номерами с 21 по 57 добавить единицу.
5. Образовать вектор $\boldsymbol{v}_1$ из 1–50 компонент вектора $\boldsymbol{w}$.
6. Образовать вектор $\boldsymbol{v}_2$ из 51–100 компонент вектора $\boldsymbol{w}$.
7. Вычислить скалярное произведение векторов $\boldsymbol{v}_1 \cdot \boldsymbol{v}_2$.
8. Вычислить поэлементное произведение векторов $\boldsymbol{v}_1$ и $\boldsymbol{v}_2$.

Задача 7.

1. Инициализировать матрицу $\boldsymbol{A} = \begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix}$.
2. Вычислить детерминант и след матрицы $\boldsymbol{A}$.
3. Вычислить выражение $\boldsymbol{B} = \boldsymbol{A}^2\boldsymbol{A}^T - \boldsymbol{A}\boldsymbol{A}^T$.
4. Вычислить детерминант и след матрицы $\boldsymbol{B}$.
5. Найти векторы $\boldsymbol{\lambda}_A$ и $\boldsymbol{\lambda}_B$ из собственных чисел матриц $\boldsymbol{A}$ и $\boldsymbol{B}$.
6. Вычислить $\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{E}$, где $\boldsymbol{E}$ – единичная матрица.
7. Вычислить $\boldsymbol{D} = \boldsymbol{B} + \boldsymbol{R}$, где $\boldsymbol{R}$ – матрица из единиц.
8. Вычислить выражение $\boldsymbol{M} = \boldsymbol{C}\boldsymbol{D}^T + \boldsymbol{A}^2 - \boldsymbol{A}^{-1}\boldsymbol{B}^3$.
9. Вычислить детерминант и след матрицы $\boldsymbol{M}$.
10. Найти собственные значения матрицы $\boldsymbol{M}$.
11. Записать второй столбец матрицы $\boldsymbol{M}$ в вектор $\boldsymbol{v}$.
12. Найти сумму и произведение компонент $\boldsymbol{v}$.

Задача 8. Решить системы линейных уравнений $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, где

1. $\boldsymbol{A} = \begin{bmatrix} 2 & 1 \\ 0 & 3 \end{bmatrix}$, $\boldsymbol{b} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$.

2. $\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 8 & 7 & 1 \end{bmatrix}$, $\boldsymbol{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$.

3. $\boldsymbol{A} = \boldsymbol{B}^{-1}\boldsymbol{C} + \left(\boldsymbol{C}\boldsymbol{D}^{-1}\right)^T + \boldsymbol{B}^2$, $\boldsymbol{b} = \boldsymbol{c} + \boldsymbol{B}\boldsymbol{d}$, где $\boldsymbol{B} = \begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix}$, $\boldsymbol{C} = \begin{bmatrix} 4 & 5 \\ 0 & 8 \end{bmatrix}$, $\boldsymbol{D} = \begin{bmatrix} 5 & 6 \\ 4 & 2 \end{bmatrix}$, $\boldsymbol{c} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $\boldsymbol{d} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$.

Задача 9. Найти собственные векторы и собственные значения матриц

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

Задача 10. Убедиться в линейной зависимости собственных векторов матрицы

$$\begin{bmatrix} 3 & 1 & 0 \\ 0 & 3 & 1 \\ 0 & 0 & 3 \end{bmatrix}.$$

Сделать это двумя способами: через вычисление детерминанта матрицы из собственных векторов и через вычисление ранга матрицы из собственных векторов.

9.2. Сетки, графики, поверхности

Задача 11. Инициализировать сетку из 10 узлов на множестве $[0, 1]$. Воспользоваться для этого функцией `linspace`.

Задача 12. Построить график функции $y(x) = \sin x$ на отрезке $[0, 2\pi]$:

1. Инициализировать сетку из $n = 10$ узлов на $[0, 2\pi]$.
2. Вычислить в каждом узле x_k сетки $y_k = \sin x_k$.
3. По полученным $\boldsymbol{x} = [x_1, \ldots, x_n]$ и $\boldsymbol{y} = [y_1, \ldots, y_n]$ построить график функции $y(x) = \sin x$.
4. Построить графики для $n = 50$, $n = 100$, $n = 1000$.

Задача 13. Проделать то же самое для функции $y(x) = e^x$ на отрезке $[0, 1]$:

1. Инициализировать сетку из $n = 10$ узлов на $[0, 1]$.
2. Вычислить в каждом узле x_k сетки $y_k = e^{x_k}$.
3. По полученным $\boldsymbol{x} = [x_1, \ldots, x_n]$ и $\boldsymbol{y} = [y_1, \ldots, y_n]$ построить график функции $y(x) = e^x$.
4. Построить графики для $n = 50$, $n = 100$, $n = 1000$.

Задача 14. Выбрав число узлов n достаточно большим, чтобы не было видно угловатостей, построить графики следующих функций:

1. $f(x) = e^{-x} \sin x$, $x \in [0, 5]$.
2. $f(x) = \dfrac{\ln x}{x^2 + 1} + |\sin x| + \sqrt[4]{1 + e^x}$, $x \in [0, 4]$.
3. $f(x) = \dfrac{1}{1 + 0.5 \cos x}$, $x \in [0, 5\pi]$.

Задача 15. Нарисовать график кусочно-заданной функции:

$$f(x) = -x,\ x \in [-1, 0], \quad f(x) = e^x - 1,\ x \in [0, 1].$$

Задача 16. Нарисовать графики следующих функций:

1. $f(x) = \begin{cases} \sqrt{1 + x^2}, & -1 \le x \le 0, \\ 2|\cos x|, & 0 < x \le 1. \end{cases}$
2. $f(x) = \begin{cases} e^{-2x} \sin x, & -1 \le x \le 0, \\ x/(1 + x^2), & 0 < x \le 1. \end{cases}$
3. $f(x) = \begin{cases} \pi \sin x, & -2\pi \le x \le -\pi, \\ \pi - |x|, & -\pi < x < \pi, \\ \pi \sin^3 x, & \pi \le x \le 2\pi. \end{cases}$

Задача 17. Нарисовать график функции $y(x) = \sin x$ на $[0, 2\pi]$ и подписать оси. Ось абсцисс подписать как x, ось ординат подписать как $y = y(x)$. Размеры шрифта выставить равными 12.

Задача 18. Нарисовать график функции $y(x) = \cos x$ на $[0, 2\pi]$, жирность линии сделать равной 2, подписать оси как в задаче 17. Размеры шрифта выставить равными 12.

Задача 19. Нарисовать график функции $y(x) = \sin x$ на $[0, 2\pi]$. В том же графическом окне нарисовать график функции $y(x) = \cos x$ на $[0, 2\pi]$, но красным цветом. Добавить легенду, подписав первый график как $\sin x$, а второй как $\cos x$. Размеры шрифта выставить равными 12.

Задача 20. В одном графическом окне отобразить следующие

зависимости. Оси подписать, жирность линий взять равным 2.0. Размер шрифта выставить равным 12. Сохранить файл с расширением `.png`, качество 300`dpi`.

1. $f_1(x) = \sin x$, $f_2(x) = \cos x$, $x \in [0, 360^\circ]$.
2. $f_1(x) = e^x$, $f_2(x) = x^2$, $f_3(x) = 1/(1+x)$, $f_4(x) = \arcsin(1/e^x)$, $x \in [0, 1]$.

Задача 21. Отобразить в одних осях графики функции

$$f(x, a) = e^{-ax} \sin ax$$

на $x \in [0, 4\pi]$ при $a = 0.01, 0.1, 1.0, 10.0$. Отобразить легенду, подписать оси. Графики нарисовать разным цветом. Жирность линии взять равной 2.0. Размеры шрифта выставить равными 12. Сохранить файл с расширением `.png`, качество 300`dpi`.

Задача 22. Отобразить зависимость $y(x) = e^{-ax} \sin ax$ на отрезке $x \in [0, 4\pi]$, в полярной системе координат (r, θ) при $a = 0.01, 0.1, 1.0, 10.0$. Графики отобразить в разных осях в одном графическом окне. Оси подписать, жирность линий взять равным 2.0. Размеры шрифта выставить равными 12. Сохранить файл с расширением `.png`, качество 300`dpi`.

Задача 23. В полярной системе координат даны зависимости

$$r(\theta) = p/(1 + e\cos\theta),$$

где $\theta \in [0, 2\pi]$, $p = 1.0$, $e = 0, 0.1, 0.2, 0.4, 0.6$. Перейти к декартовой системе координат (x, y) и отобразить эти зависимости на плоскости (x, y) в одних осях. Оси подписать, жирность линий взять равным 2.0. Размеры шрифта выставить равными 12. Сохранить файл с расширением `.png`, качество 300`dpi`.

Задача 24. Нарисовать в одном графическом окне, но *в разных осях* графики функций $y(x) = \sin x$ и $y(x) = \cos x$. Все оси подписать.

Задача 25. Инициализировать равномерную сетку на множестве $[0, 1] \times [0, 1]$ с помощью функции `meshgrid`. По вертикальной оси взять $m = 5$ точек, а по горизонтальной оси взять $n = 10$ точек.

Задача 26. Взяв достаточно плотные сетки, нарисовать поверхности и контуры следующих функций:

1. $f(x, y) = x^2 + y^2$, $x \in [-1, 1]$, $y \in [-1, 1]$.
2. $f(x, y) = x^3 + y^3$, $x \in [-1, 1]$, $y \in [-1, 1]$.

3. $f(x, y) = x \exp(-x^2 - y^2)$, $x \in [-1, 1]$, $y \in [-1, 1]$.
4. $f(x, y) = \sin x \cos y$, $x \in [-2\pi, 2\pi]$, $y \in [-2\pi, 2\pi]$.

9.3. Работа с функциями

Задача 27. Создать функцию `func`, которая ничего не принимает на вход, ничего не выдает и ничего не делает.

Задача 28. Создать функцию `func`, которая принимает на вход число x, а выдает $y = x^2$. Вызвать функцию, подавая на вход разные векторы, константы, матрицы и объяснить полученные результаты/ошибки.

Задача 29. Создать функцию `func`, которая принимает на вход число a и число b (то есть два аргумента), а выдает число $c = a + b$. Вызвать функцию, подавая на вход разные векторы, константы, матрицы и объяснить полученные результаты/ошибки.

Задача 30. Создать функцию `func`, которая принимает на вход *один* аргумент – вектор $\boldsymbol{x}$, а выдает сумму его компонент. Заметим, что это не то же самое, что функция с *несколькими* аргументами, как в предыдущей задаче. Вызвать функцию, подавая на вход разные векторы, константы, матрицы и объяснить полученные результаты/ошибки.

Задача 31. Создать функцию, которая ничего не принимает на вход, но выдает два аргумента: единичную матрицу размера 3×3 и нулевую матрицу размера 3×3.

Задача 32. Создать функцию, которая на вход принимает два аргумента: переменную t и вектор $\boldsymbol{y} = [y_1, y_2]$ с двумя компонентами, а на выходе выдает вектор с двумя компонентами $[t+y_1, t^2+y_2]$. Вызвать эту функцию разными командами, подавая на вход первого аргумента константы, а на вход второго аргумента векторы, константы, матрицы. Попробовать подавать на вход первого аргумента векторы и матрицы, объяснить полученные результаты/ошибки.

Задача 33. Создать функцию, которая принимает на вход два аргумента: переменную t и вектор $\boldsymbol{y} = [y_1, y_2]$ с двумя компонентами, а на выходе выдает вектор с двумя компонентами $[y_2, -y_1]$. Переменная t в теле функции не используется. Вызвать функцию, подавая на вход разные векторы, константы, матрицы и объяснить полученные результаты/ошибки. Изменить значение какой-либо компоненты выходной переменной и проверить, изменяется ли значение входной

переменной. Результат объяснить. Добиться того, чтобы изменение одной переменной не влияло на другую.

Задача 34. Создать функцию, которая принимает на вход два аргумента: переменную t и вектор $\boldsymbol{y}$, а на выходе выдает вектор $\boldsymbol{y}/|\boldsymbol{y}|^3$. Переменная t в теле функции не используется. Вызвать функцию, подавая на вход разные векторы, константы, матрицы и объяснить полученные результаты/ошибки. Изменить значение какой-либо компоненты выходной переменной и проверить, изменяется ли значение входной переменной. Результат объяснить. Добиться того, чтобы изменение одной переменной не влияло на другую.

Задача 35. Создать функцию, которая принимает на вход два аргумента: переменную t и вектор $\boldsymbol{y}$ размерности 6, а на выходе выдает два вектора: $\boldsymbol{r}$, который содержит первые три компоненты вектора $\boldsymbol{y}$, и $\boldsymbol{v}$, который содержит последние три компоненты вектора $\boldsymbol{y}$. Другими словами, вектор $\boldsymbol{y}$ имеет известную структуру $\boldsymbol{y} = [\boldsymbol{r}, \boldsymbol{v}]$, и действие функции заключается в том, чтобы «вытащить» из $\boldsymbol{y}$ подвектор $\boldsymbol{r}$ и подвектор $\boldsymbol{v}$. Изменить значение какой-либо компоненты выходной переменной и проверить, изменяется ли значение выходной переменной. Результат объяснить. Добиться того, чтобы изменение одной переменной не влияло на другую.

Задача 36. Создать функцию, которая принимает на вход три аргумента: переменную t, шестимерный вектор $\boldsymbol{y} = [\boldsymbol{r}, \boldsymbol{v}]$, где $\boldsymbol{r}$ и $\boldsymbol{v}$ – трехмерные подвекторы, и переменную μ. Функция должна выдавать шестимерный вектор $\boldsymbol{z} = [\boldsymbol{v}, -\mu\boldsymbol{r}/|\boldsymbol{r}|^3]$. Эта функция пригодится при реализации уравнений движения в задаче двух тел. Как и в предыдущих примерах, убедиться, что входные и полученные выходные переменные не связаны изменениями.

Задача 37. Создать функцию, которая принимает на вход матрицу $\boldsymbol{M}$ размера 3×3, а выдает один вектор с двумя элементами: $[M_{12} + M_{21}, M_{11} - M_{22}]$. Вызвать эту функцию, подавая разные матрицы 3×3, 2×2, попробовать подать векторы, константы. Объяснить полученные результаты/ошибки.

Задача 38. Создать функцию, которая принимает на вход матрицу $\boldsymbol{M}$ размера 2×2, а выдает вектор $[M_{11}, M_{21}, M_{12}, M_{22}]$. При решении использовать функцию `reshape`. Вызвать эту функцию, подавая разные матрицы 2×2, 3×3. Объяснить, почему не возникает ошибок при подаче матриц 3×3. Добиться, чтобы после получения результата выходная и входная переменные не были связаны изменениями.

Задача 39. Создать функцию, которая принимает на вход матрицу $\boldsymbol{M}$ размера 2×2, далее по этой матрице создает векторы $\boldsymbol{v} = [M_{11}, M_{21}, M_{12}, M_{22}]$ и $\boldsymbol{w} = [M_{11}, M_{12}, M_{21}, M_{22}]$ и выдает на выходе сумму этих векторов. При создании векторов $\boldsymbol{v}$ и $\boldsymbol{w}$ использовать функцию `reshape`. Проверить, что после получения результата выходная и входная переменные не связаны изменениями.

9.4. Циклы и логическое управление

Задача 40. Создать `if`-окружения, которые будут проверять значение переменной a и выдавать соответствующее значение b:

1) если $a < 0$, то $b = -1$, иначе $b = 1$;
2) если $a < 0$, то $b = -1$, если $a = 0$, то $b = 0$, а если $a > 0$, то $b = 1$;
3) если $a \in (1, 2)$, то $b = 1$, иначе $b = 0$;
4) если $a \leq 0$ и $a > -2$, то $b = 1$, иначе $b = 0$;
5) если $a \leq 0$ или $a > 3$, то $b = 1$, иначе $b = 0$;
6) если $a \in (0, 2]$, то $b = 1$, если $a \in (2, 10)$, то $b = 0$, иначе $b = -1$.

Задача 41. С помощью циклов `for`

1) увеличить последовательно значение переменной S с 1 до 10;
2) посчитать сумму $S = 1 + 2 + \cdots + 1000$;
3) посчитать сумму $S = 1^1 + 2^2 + \cdots + 6^6$;
4) инициализировать вектор $\boldsymbol{a}$ с компонентами $a_k = e^k$, $k = 0 \ldots 4$;
5) рекуррентно вычислить a_{10} по формуле $a_k = a_{k-1} + k$, считая $a_0 = 0$;
6) вычислить a_{10} по формуле $a_k = a_{k-1} + a_{k-2}$, считая $a_0 = 1$, $a_1 = 1$. Это последовательность Фибоначчи;
7) вычислить a_{10} по формуле $a_k = 0.5 \cdot (a_{k-1} + 2/a_{k-1})$, $a_0 = 1$. Вообще $a_k = 0.5 \cdot (a_{k-1} + x/a_{k-1})$ сходится к $\sqrt{x}$ при $k \to \infty$, это так называемая формула Герона;
8) вычислить a_{10} по формуле $a_k = (\sqrt{2})^{a_{k-1}}$, $a_0 = \sqrt{2}$. Можно доказать, что a_k сходится к 2 при $k \to \infty$.

Задача 42. С помощью циклов `while`

1) увеличивать S с нуля на единицу до тех пор, пока S не превысит 5;

2) считать a_k по формуле $a_k = 0.5 \cdot (a_{k-1} + 2/a_{k-1})$, $a_0 = 1$, до тех пор, пока не будет выполнено $|a_k - \sqrt{2}| < 10^{-10}$. Найти минимальный k, когда это неравенство выполнено;

3) считать a_k по формуле $a_k = (\sqrt{2})^{a_{k-1}}$, $a_0 = \sqrt{2}$, до тех пор, пока не будет выполнено $|a_k - 2| < 10^{-10}$. Найти минимальный k, когда это неравенство выполнено.

9.5. Простейшие численные методы

Задача 43. Решить уравнение $f(x) = \sin x = 0$ методом Ньютона. Для этого взять $x_0 = 1$ и итерационно с помощью цикла `while` вычислять

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{\sin x_k}{\cos x_k}$$

до тех пор, пока не будет выполнено $|\sin x_k| < 10^{-10}$. Попробовать взять $x_0 = 3$ и посмотреть, к чему сойдется в этот раз. Объяснить результат. Почему метод Ньютона не работает для $x_0 = \pi/2$?

Задача 44. Решить методом Ньютона уравнения. Для этого сначала нарисовать графики входящих сюда функций, визуально подобрать подходящие x_0 и с них стартовать итерации по методу Ньютона.

1. $\cos x = 0$.
2. $x - e \sin x = 1$, где $e = 0.1$ (уравнение Кеплера).
3. $e^x = 5 - x$.
4. $\ln x = x - 3$.

Задача 45. Реализовать метод дихотомии и решить им уравнения ниже. Метод дихотомии состоит в следующем. Дано уравнение $f(x) = 0$ с непрерывной и монотонной функцией $f(x)$, причем известно, что $f(a)f(b) < 0$, то есть знаки функции на концах отрезка $[a, b]$ разные. Возьмем среднюю точку $c = (a + b)/2$ и вычислим $f(c)$. Если $f(a)f(c) < 0$, то решение находится на отрезке $[a, c]$, поэтому присваиваем $b = c$ и все повторяем. Если же $f(a)f(c) > 0$, то решение находится на отрезке $[c, b]$, поэтому присваиваем $a = c$ и все повторяем. Повторять все это будем до тех пор, пока разность $|a - b|$ не станет достаточно малой, либо пока $\min(|f(a)|, |f(b)|)$ не станет достаточно малым. Для решения этой задачи нужно воспользоваться циклом `while` и окруже-

нием `if`. Метод дихотомии выделить в отдельную функцию, чтобы ей можно было воспользоваться в других проектах.

1. $\sin x = 0$, $[a, b] = [-1, 1]$.
2. $e^x = 5 - x$, $[a, b] = [1, 2]$.
3. $\ln x = x - 3$, $[a, b] = [3, 6]$.

Задача 46. Оценить производную функции $f(x) = \sin x$ в точке $x_0 = 0$ по формуле

$$f'(x_0) \approx f'_h(x_0) = \frac{f(x_0 + h) - f(x_0)}{h}.$$

Взять $h = 10^{-1}$, 10^{-2} и так далее до $h = 10^{-20}$. Построить график по точкам $(-\log_{10} h, f'_h - 1)$, чтобы посмотреть, как изменяется погрешность определения производной с ростом порядка h.

Задача 47. Теперь оценить производную функции $f(x) = \sin x$ в точке $x_0 = 0$ по формуле

$$f'(x_0) \approx f'_h(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h}.$$

Взять $h = 10^{-1}$, 10^{-2} и так далее до $h = 10^{-20}$. Построить график по точкам $(-\log_{10} h, f'_h - 1)$, чтобы посмотреть, как меняется оценка производной с ростом порядка h. Сравнить результаты с результатами предыдущей задачи. Где сходимость к истинному значению (единице) оказывается быстрее? Обосновать формально.

Задача 48. Пусть известны значения функции $f(x_0)$ в точке x_0 и $f(x_1)$ в точке x_1. Метод линейной интерполяции позволяет оценить значение функции $f(x)$ в точке $x \in [x_0, x_1]$ по формуле

$$f(x) \approx \frac{x - x_0}{x_1 - x_0} f(x_1) + \frac{x_1 - x}{x_1 - x_0} f(x_0).$$

Создать функцию `LinInterp`, которая будет принимать на вход числа x_0, x_1, $f(x_0)$, $f(x_1)$, x и выдавать приближенное значение $f(x)$ по формуле выше. Воспользоваться функцией `LinInterp` в следующих ситуациях. Во всех случаях сравнить полученное значение с истинным значением функции в точке:

1. $f(x) = x^2$, $x_0 = 1$, $x_1 = 3$, $x = 2$.
2. $f(x) = \sin x$, $x_0 = 0$, $x_1 = \pi/2$, $x = \pi/4$.
3. $f(x) = \ln(1 + x)$, $x_0 = 0$, $x_1 = 10$, $x = 5$.

9.6. Интегрирование ОДУ

Задача 49. Рассмотрим дифференциальное уравнение $y' = y$, $y(0) = 1$. Требуется решить его численно методом Эйлера на отрезке $[0, 1]$. Для этого взять $h = 0.01$ и итерационно с помощью цикла `for` вычислить

$$y_{k+1} = y_k + hy_k,\ k = 1, \ldots, N,\ N = 1/h.$$

Отобразить на графике истинное решение $y(t) = e^t$ и найденное приближенное в точках (t_k, y_k), $t_k = h(k-1)$, $k = 1, \ldots, N+1$.

Задача 50. Решить методом Эйлера уравнение $y' = y$, $y(0) = 1$ для шага $h_1 = 0.01$ и шага $h_2 = 0.001$. Для каждого из этих двух случаев вычислить глобальную ошибку $\Delta = |y_{N+1} - e|$, получить соответственно Δ_1 и Δ_2. Вычислить отношение Δ_2/Δ_1. Так как метод Эйлера имеет порядок 1, то это отношение будет равно примерно $h_2/h_1 = 0.1$. Обосновать это формально.

Задача 51. Проделать то же самое, что в предыдущей задаче, но для уравнений $y' = -y$, $y(0) = 1$ и $y' = t + y$, $y(0) = 1$ на отрезке $[0, 1]$.

Задача 52. Реализовать классический метод Рунге – Кутты четвертого порядка. Для этого создать отдельную функцию `RK4(func, t0, tf, y0, h)`, которая принимает на вход указатель на функцию правых частей уравнений `func`, начальный момент времени `t0`, конечный момент времени `tf`, начальный вектор зависимых переменных `y0` и шаг на оси времени `h`. Предполагается, что `func` принимает на вход только два аргумента – время `t` и вектор зависимых переменных `y`. Функция `RK4` должна выдавать два массива: `T`, который содержит узлы сформированной сетки, и `Y` – матрицу, каждый столбец (или строка) которой равен приближенному решению $\boldsymbol{y}$ в соответствующем узле.

Задача 53. Проинтегрировать классическим методом Рунге – Кутты четвертого порядка уравнения из задачи 50 и 51 и сравнить на графиках точные и приближенные решения.

Задача 54. Создать функцию `EqIncreasingExponent(t,y)`, которая будет принимать на вход переменную t, переменную y и будет выдавать y. Это будет функция правых частей уравнения $y' = y$, решением которого являются функции вида Ce^t. Проинтегрировать полученные уравнения движения с начальным условием $y(0) = 1$ методами Дормана – Принса 5(4) и LSODA (или любым другим многошаговым методом с адаптивным шагом и переменного порядка). Сравнить на графиках точное и приближенные решения.

Задача 55. Создать функцию `EqDecreasingExponent(t,y)`, которая будет принимать на вход переменную t, переменную y и будет выдавать $-y$. Это будет функция правых частей уравнения $y' = -y$, решением которого являются функции вида Ce^{-t}. Проинтегрировать полученные уравнения движения с начальным условием $y(0) = 1$ методом Дормана – Принса 5(4) и LSODA (или любым другим многошаговым методом с адаптивным шагом и переменного порядка). Сравнить на графике точное и приближенные решения.

Задача 56. Создать функцию `EqHarmonicOscillator(t,y)`, которая будет принимать на вход переменную t, вектор $\boldsymbol{y} = [y_1, y_2]$ и будет выдавать вектор $[y_2, -y_1]$. Это будет функция правых частей системы уравнений $y_1' = y_2$, $y_2' = -y_1$ (гармонический осциллятор), решением которой являются функции вида $y_1 = A\sin t + B\cos t$, $y_2 = A\cos t - B\sin t$. Проинтегрировать полученные уравнения движения с начальным условием $\boldsymbol{y}_0 = [0, 1]$ методом Дормана – Принса 5(4) и LSODA (или любым другим многошаговым методом с адаптивным шагом и переменного порядка). Сравнить на графиках точное и приближенные решения.

Задача 57. Проинтегрировать уравнения $y' = y$, $y(0) = 1$ классическим методов Рунге – Кутты четвертого порядка на таком интервале времени, чтобы в конечный момент времени было $y(T) = e^2$. Для этого на уровне интегратора реализовать выход на ограничение с помощью метода дихотомии. В результате должно получиться число $T \approx 2$.

Задача 58. Решить предыдущую задачу, но для выхода на ограничение воспользоваться методом интерполяции четвертого порядка. Реализовать метод интерполяции можно самостоятельно, а можно воспользоваться готовыми решениями из математических пакетов. Сравнить скорость и точность процесса выхода на ограничение при использовании метода дихотомии и метода интерполяции.

9.7. Моделирование орбитального движения

Задача 59. Реализовать функции преобразования истинной аномалии в эксцентрическую аномалию и обратно.

Задача 60. Реализовать функцию `SolveKeplerEq(M, e)`, которая принимает на вход среднюю аномалию M, эксцентриситет e, а выдает решение уравнения Кеплера:

$$E - e\sin E = M$$

относительно эксцентрической аномалии E.

Задача 61. Создать функцию `r2bp(t,y,mu)`, которая будет принимать на вход переменную t, вектор $\boldsymbol{y} = [\boldsymbol{r}, \boldsymbol{v}]$ с шестью компонентами, из которых первые три компоненты $\boldsymbol{r}$ – радиус-вектор космического аппарата, а другие три компоненты $\boldsymbol{v}$ – скорость аппарата, а также число μ – гравитационный параметр притягивающего центра. Функция эта будет выдавать вектор $[\boldsymbol{v}, -\mu\boldsymbol{r}/|\boldsymbol{r}|^3]$, то есть правую часть уравнений задачи двух тел. Проинтегрировать уравнения движения с начальным условием $\boldsymbol{y}_0 = [\boldsymbol{r}_0, \boldsymbol{v}_0]$, $\boldsymbol{r}_0 = [1, 0, 0]$, $\boldsymbol{v} = [0, 1, 0]$, и гравитационным параметром $\mu = 1$, на участке времени $t \in [0, 2\pi]$ методом Дормана – Принса 5(4). Сравнить с точным решением (период круговой орбиты единичного радиуса).

Задача 62. Реализовать функцию `rv2oe(rv, mu)`, которая принимает на вход шестимерный вектор `rv`, содержащий радиус-вектор и скорость аппарата, гравитационный параметр `mu`, а выдает вектор классических орбитальных элементов `oe`, содержащий большую полуось a, эксцентриситет e, наклонение i, долготу восходящего узла Ω, аргумент перицентра ω и истинную аномалию ϑ.

Задача 63. Реализовать функцию `oe2rv(oe, mu)`, которая принимает на вход шестимерный вектор `oe`, содержащий классические орбитальные элементы (см. предыдущую задачу), и гравитационный параметр `mu`, а выдает вектор `rv`, содержащий радиус-вектор и скорость аппарата.

Задача 64. Проверить работу функции `oe2rv`. Для этого рассмотреть векторы `oe`, которые дают известные и понятные значения элементов орбиты. Например, можно взять `oe` равным $[a, e, i, \Omega, \omega, \vartheta] =$ $= [1, 0, 0, 0, 0, 0]$ и получить радиус-вектор $[1, 0, 0]$ и скорость $[0, 1, 0]$.

Задача 65. Проверить согласованность функций `rv2oe` и `oe2rv`, то есть равенства

```
oe = rv2oe(oe2rv(oe, mu), mu),
rv = oe2rv(rv2oe(rv, mu), mu).
```

Для этого

1) задать произвольно вектор `rv0` и гравитационный параметр `mu`;
2) рассчитать для `rv0` и `mu` вектор элементов `oe0`;
3) рассчитать для `oe0` и `mu` вектор `rv1`;
4) рассчитать для `rv1` и `mu` вектор `oe1`;
5) сравнить векторы `rv0` и `rv1`, а также `oe0` и `oe1`. С высокой точностью сравниваемые векторы должны совпадать.

Вектор `rv0` и число `mu` можно формировать с использованием псевдослучайных чисел, а сравнение векторов можно производить в цикле с большим числом итераций.

Задача 66. Реализовать функцию `rv2ee(rv, mu)`, которая принимает на вход шестимерный вектор `rv`, содержащий радиус-вектор и скорость аппарата, гравитационный параметр `mu`, а выдает вектор равноденственных орбитальных элементов `ee`, содержащий элементы p, e_x, e_y, i_x, i_y, L.

Задача 67. Реализовать функцию `ee2rv(ee, mu)`, которая принимает на вход шестимерный вектор `ee`, содержащий равноденственные орбитальные элементы (см. предыдущую задачу), и гравитационный параметр `mu`, а выдает вектор `rv`, содержащий радиус-вектор и скорость аппарата.

Задача 68. Проверить работу функции `ee2rv`. Для этого рассмотреть векторы `ee`, которые дают известные и понятные значения элементов орбиты. Например, можно взять `ee` равным $[p, e_x, e_y, i_x, i_y, L] = [1, 0, 0, 0, 0, 0]$ и получить радиус-вектор $[1, 0, 0]$ и скорость $[0, 1, 0]$.

Задача 69. Проверить согласованность функций `rv2ee` и `ee2rv` так же, как это предлагается сделать в задаче 65.

Задача 70. Реализовать функцию `rv2hcat(rv, mu)`, которая переводит положение и скорость аппарата в вектор с компонентами h (энергия аппарата), $\boldsymbol{c}$ (орбитальный момент аппарата), $\boldsymbol{A}$ (вектор Лапласа) и τ (время полета от перицентра до текущего положения).

Задача 71. Реализовать функцию `hcat2rv(hca, mu)`, которая переводит вектор с компонентами h (энергия аппарата), $\boldsymbol{c}$ (орбитальный момент аппарата), $\boldsymbol{A}$ (вектор Лапласа) и τ (время полета от перицентра до текущего положения) в вектор, содержащий положение и скорость аппарата.

Задача 72. Проверить согласованность функций `rv2hcat` и `hcat2rv` так же, как это предлагается сделать в задаче 65.

Задача 73. Реализовать функцию `ine2orb(xine, rv)`, которая принимает на вход трехмерный или шестимерный вектор `xine`, заданный в инерциальной системе координат, и шестимерный вектор `rv`, содержащий положение и скорость аппарата в инерциальной системе координат, и выдает вектор `xorb` в орбитальной системе координат.

Задача 74. Реализовать функцию `orb2ine(xorb, rv)`, которая принимает на вход трехмерный или шестимерный вектор `xorb`, за-

данный в орбитальной системе координат, и шестимерный вектор `rv`, содержащий положение и скорость аппарата в инерциальной системе координат, а выдает вектор `xine` в инерциальной системе координат.

Задача 75. Проверить согласованность функций `ine2orb` и `orb2ine` так же, как это предлагается сделать в задаче 65.

Задача 76. Реализовать уравнения в равноденственных элементах. Для этого реализовать функцию `EqEquinoctial(t, ee, mu)`, которая принимает на вход время `t`, вектор равноденственных элементов `ee`, гравитационный параметр `mu`, а выдает правую часть уравнений в равноденственных элементах.

Задача 77. Реализовать функцию `fj2(rv, mu)`, которая принимает на вход вектор, содержащий положение и скорость аппарата, гравитационный параметр, а выдает возмущающее ускорение за счет действия земной гармоники J_2.

Задача 78. Реализовать функцию `fgrav(rsc, rbody, mu_body)`, которая принимает на вход радиус-вектор аппарата `rsc`, радиус-вектор небесного тела `rbody`, гравитационный параметр небесного тела `mu_body` и выдает возмущающее ускорение за счет гравитационного притяжения небесным телом аппарата.

Задача 79. Реализовать функцию `fsrp(adm, rsc, rcent, rsun, Rcent, Rsun)`, которая принимает на вход отношение площади к массе аппарата `adm`, радиус-вектор аппарата `rcent`, радиус-вектор Солнца `rsun`, радиус центрального тела `Rcent`, радиус Солнца `Rsun`, а выдает возмущающее ускорение за счет давления солнечного излучения с учетом тени (принять цилиндрическую модель тени без «полутеней»).

Задача 80. Реализовать уравнения в вариациях в задаче двух тел. А именно, создать функцию `r2bp_vareqs(t, z, mu)`, которая принимает на вход время `t`, вектор `z`, содержащий радиус-вектор $\boldsymbol{r}$, скорость $\boldsymbol{v}$ и записанную в виде вектора матрицу перехода размера 6×6, и гравитационный параметр. Функция должна возвращать вектор, который будет состоять из правой части уравнений задачи двух тел и уравнения в вариациях (уравнений для $\partial \boldsymbol{y}/\partial \boldsymbol{y}_0$, где $\boldsymbol{y} = [\boldsymbol{r}, \boldsymbol{v}]$). Переходы к векторному представлению матрицы перехода и обратно осуществлять с помощью функции `reshape`.

Задача 81. Для каких-либо начальных условий проинтегрировать уравнения в вариациях в модели задачи двух тел *на периоде орбиты* T, найти матрицу перехода $\Phi(T)$ (она называется матрицей монодромии). Найти ее собственные значения (мультипликаторы Флоке).

Задача 82. С помощью процедуры выхода на ограничение проинтегрировать уравнения движения в задаче двух тел с начальным условием $\boldsymbol{r}_0 = [1, 0, 0]$ и $\boldsymbol{v}_0 = [0, 1.1, 0]$, единичным гравитационным параметром, на таком участке времени, чтобы в конечный момент времени аппарат оказался на оси ординат. Для выхода на ограничение воспользоваться методом дихотомии и методом интерполяции четвертого порядка, который можно реализовать самостоятельно или взять готовое решение из математического пакета. Сравнить точность и скорость выхода на ограничение этими двумя методами.

Задача 83. Реализовать уравнения Шперлинга – Боде. А именно, реализовать функцию `EqSperlingBurdet(s, y, mu)`, которая принимает на вход фиктивное время `s`, вектор `y`, содержащий радиус-вектор, расстояние до притягивающего центра, скорость, энергию, вектор Лапласа и физическое время, и гравитационный параметр `mu`. Выбрать в качестве начального условия вектор, соответствующий $\boldsymbol{r}_0 = [1, 0, 0]$ и $\boldsymbol{v} = [0, 1, 0]$, единичный гравитационный параметр, и проинтегрировать уравнения движения на участке $s \in [0, 2\pi]$. В результате должен получиться один виток вдоль круговой орбиты единичного радиуса.

Задача 84. Реализовать функцию `rv2ks(rv, mu)`, которая переводит положение и скорость аппарата в вектор с компонентами $\boldsymbol{u}$ и $\boldsymbol{u}'$ – векторами Кустаанхеймо – Штифеля.

Задача 85. Реализовать функцию `ks2rv(ks, mu)`, которая переводит вектор с компонентами векторов Кустаанхеймо – Штифеля в вектор, содержащий положение и скорость аппарата.

Задача 86. Проверить согласованность функций `rv2ks` и `ks2rv` так же, как это предлагается сделать в задаче 65.

Задача 87. Реализовать уравнения Кустаанхеймо – Штифеля. А именно, реализовать функцию `EqKustaanheimoStiefel(s, y, mu)`, которая принимает на вход фиктивное время `s`, вектор `y`, содержащий переменные Кустаанхеймо – Штифеля, расстояние до притягивающего центра, энергию и физическое время, и гравитационный параметр `mu`. Выбрать в качестве начального условия вектор, соответствующий $\boldsymbol{r}_0 = [1, 0, 0]$ и $\boldsymbol{v} = [0, 1, 0]$, единичный гравитационный параметр, и проинтегрировать уравнения движения на участке $s \in [0, 2\pi]$. В результате должен получиться один виток вдоль круговой орбиты единичного радиуса в декартовых координатах и половина витка в переменных Кустаанхеймо – Штифеля.

9.8. Проектирование орбит

Задача 88. Реализовать функцию `residue(v0, t0, tf, r0, rf)`, которая принимает на вход начальную скорость аппарата `v0`, начальный момент времени `t0`, конечный момент времени `tf`, начальный радиус-вектор `r0`, конечный радиус-вектор `rf`, далее интегрирует уравнения движения в модели задачи двух тел на интервале времени от `t0` до `tf` с начальными условиями `r0` и `v0` и выдает разность (невязку, residue) между полученным радиусом-вектором в конечный момент времени и вектором `rf`.

Задача 89. Пусть $\boldsymbol{r}_0 = [1, 0, 0]$, $\boldsymbol{r}_f = [0, 1.2, 0]$, $t_0 = 0$, $t_f = \pi/2$, $\mu = 1$. Найти начальную скорость v_0 и соответствующую траекторию, которая переведет аппарат из положения $\boldsymbol{r}_0$ в $\boldsymbol{r}_f$ за время от t_0 до t_f в модели задачи двух тел. Для этого воспользоваться функцией `residue` из предыдущей задачи и каким-либо готовым численным методом решения алгебраических уравнений, способным самостоятельно оценивать якобиан оптимизируемой функции. В качестве начального приближения взять $\boldsymbol{v}_0 = [0, 1, 0]$. На каждой итерации выводить номер итерации, число обращений к оптимизируемой функции, норму невязки. Зарегистрировать время расчетов.

Задача 90. Повторить предыдущую задачу, но в этот раз реализовать вычисление якобиана оптимизируемой функции с использованием уравнений в вариациях. При тех же условиях зарегистрировать и сравнить число итераций, обращений к оптимизируемой функции и общее время расчета.

Задача 91. Повторить ту же задачу, но решить ее теперь методом параллельной пристрелки, сделав три участка траектории на пути к точке-цели. Для этого следует скорректировать функцию `residue`, теперь там должен появиться цикл по участкам, на них будут интегрироваться уравнения движения и записываться результаты интегрирования. На каждом участке начальным условием для матрицы перехода будет служить единичная матрица. Вычислить все невязки в соответствии с методом параллельной пристрелки и записать их в один выходной из функции вектор.

Задача 92. Реализовать функцию `objective`, которая принимает на вход и делает то же самое, что указанная в задаче 88 функция `residue`, но выдает норму вектора невязок (целевая функция). Минимизировать эту функцию градиентным методом спуска, а также безградиентными методами Нелдера – Мида и роя частиц. Сравнить результаты с результатами оптимизации вектора невязок.

Заключение

В пособии приведен семестровый курс лекций по «Численным методам механики космического полета». Рассматриваются методы общего назначения, зарекомендовавшие себя в механике полета (методы решения алгебраических и дифференциальных уравнений, методы оптимизации), а также методы, специфические для предметной области (регуляризация и стабилизация уравнений движения, выбор переменных движения). Предложены задачи для самостоятельного решения, которые можно решать на разных языках программирования. Литература в конце пособия рекомендуется для углубления и расширения знаний в вышеуказанных разделах численных методов.

Список литературы

1. Авдюшев В.А. Численное моделирование орбит. Томск : НТЛ, 2010.
2. Бордовицына Т.В., Авдюшев В.А. Теория движения искусственных спутников Земли. Аналитические и численные методы : учеб. пособие. Томск : Томский университет, 2007.
3. Григорьев И.С. Методическое пособие по численным методам решения краевых задач принципа максимума в задачах оптимального управления. Москва : Изд-во Центра прикл. исслед. при мех.-мат. факультете МГУ, 2005.
4. Иванов Д.С., Трофимов С.П., Широбоков М.Г. Численное моделирование орбитального и углового движения космических аппаратов / под ред. М. Ю. Овчинникова. Москва : ИПМ им. М.В. Келдыша, 2016.
5. Калиткин Н.Н. Численные методы : учеб. пособие. 2-изд. Санкт-Петербург : БХВ-Петербург, 2014.
6. Константинов М.С., Петухов В.Г., Тейн М. Оптимизация траекторий гелиоцентрических перелетов. Москва : МАИ, 2015.
7. Поляк Б.Т. Введение в оптимизацию. Москва : ЛЕНАНД, 2021.
8. Самохин А.С. Методика построения экстремалей Понтрягина в задачах сквозной траекторной оптимизации межпланетных перелётов с учётом планетоцентрических участков. Дис. ... канд. физ.-мат. наук. Москва : МГУ, 2020.
9. Суханов А.А. Астродинамика. Москва : Институт космических исследований РАН, 2010.
10. Сухарев А.Г., Тимохов А.В., Федоров В.В. Курс методов оптимизации : учеб. пособие. 2 изд. Москва : ФИЗМАТЛИТ, 2008.

11. Хайрер Э., Нёрсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи: пер. с англ. Москва : Мир, 1990.

12. Холл Дж., Уатт Дж. Современные численные методы решения обыкновенных дифференциальных уравнений. Москва : Мир, 1979.

13. Штифель Е., Шейфеле Г. Линейная и регулярная небесная механика. Москва : Наука, 1975.

14. Прикладная небесная механика и управление движением. Сборник статей, посвященных 90-летию со дня рождения Д.Е. Охоцимского / составители: Т.М. Энеев, М.Ю. Овчинников, А.Р. Голиков. Москва : ИПМ им. М.В. Келдыша, 2010.

15. Nocedal J., Wright S.J. Numerical optimization. Springer, 2006.

16. Simon D. Evolutionary optimization algorithms. Wiley, 2013.