МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Нижегородский государственный университет им. Н.И.Лобачевского
Радиофизический факультет

ЗАДАЧА КОШИ ДЛЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ

Лабораторная работа для студентов радиофизического факультета ННГУ

Нижний Новгород 2002г.

УДК 518.5 Безусловный экстремум. Введение в численные методы. /сос. В.В.Кулинич, И.П.Смирнов - Нижний Новгород, изд. ННГУ им. Н.И.Лобачевского, 2002 г. -32 с.

Излагаются основные методы построения разностных схем численного решения задачи Коши для обыкновенных дифференциальных уравнений, вопросы их сходимости и устойчивости. Приводится полный текст программы на алгоритмическом языке Фортран решения задачи Коши методом Рунге-Кутты. Формулируются задания для самостоятельной работы по вычислительному практикуму.

Методическая разработка рассчитана на студентов и магистрантов радиофизического факультета Нижегородского госуниверситета.

Составители:
Виктор Валентинович Кулинич
Иван Паисьевич Смирнов

Нижегородский государственный университет им.Н.И.Лобачевского
2002 г.

1 Основные методы построения разностных схем для задачи Коши

1.1 Введение

Широкие классы задач физики, химии, биологии, экономики и др. сводятся к решению начально-краевых задач для систем обыкновенных дифференциальных уравнений. Считается, что приведение задачи к форме, например, задачи Коши является гарантией ее решения, хотя бы приближенного. Теория численного решения таких задач является одной из развитых областей вычислительной математики.

Нашей целью является обзор основных численных методов решения задачи Коши

$$\begin{cases} y'(x) = f(x, y(x)),\ x \in (a, b) \\ y(a) = y^0, \end{cases} \tag{1}$$

где $y(x) \equiv (y_1(x), y_2(x), \ldots, y_s(x)),\ y^0$ - заданный начальный вектор.

При численном решении задачи (1) ищется последовательность векторов $\{y_n\}_{n=0}^N$, являющихся приближениями для значений решения $\{y(x_n)\}_{n=0}^N$ на множестве точек сетки $\omega_N \equiv \{x_i :\ x_{i+1} = x_i + h_i,\ i = \overline{0, N-1},\ x_0 = a,\ x_N = b\}$, где $h_i > 0$ - шаг сетки. В практике используются разнообразные сетки. Мы ограничимся рассмотрением методов с равномерными сетками, т.е. сетками с постоянным шагом $h_i \equiv h$. Основное предположение относительно задачи (1), которое мы примем, состоит в непрерывности правой части дифференциального уравнения (вектор-функции $f(x, y)$) по совокупности переменных в $(a, b) \times R^s$ и глобального (равномерного относительно x) условия Липшица по y:

$$\|f(x, y^1) - f(x, y^2)\| \le L \left\|y^1 - y^2\right\|,$$

где L - постоянная Липшица. В этих условиях решение задачи Коши (1) существует и единственно при любом начальном векторе y^0 (см. [1]). Заметим, что реализация конкретного численного метода и, в особенности, его обоснование требуют обычно более сильных ограничений на параметры задачи. Не оговаривая дополнительно, мы предполагаем всякий раз выполнение этих условий, ограничиваясь в основном изложением формальной схемы метода.

Мы также "по умолчанию" для простоты ограничиваемся рассмотрением скалярного случая.

1.2 Метод разложения в ряд Тейлора

Простейшим способом построения приближенного решения в точке x_{n+1} сетки ω_N является способ, основанный на разложении решения в ряд Тейлора в предыдущей точке сетки x_n по степеням шага h:

$$y(x_{n+1}) = y(x_n) + h\triangle(x_n, y_n, h), \tag{2}$$

$$\triangle(x, y, h) \equiv y'(x) + \frac{h}{2}y''(x) + \frac{h^2}{3!}y'''(x) + \dots.$$

Взяв вместо этого ряда конечный его отрезок

$$\varphi_p(x, y, h) \equiv y'(x) + \frac{h}{2}y''(x) + \ldots + \frac{h^{p-1}}{p!}y^{(p)}(x)$$

и заменяя в нем производные $y^{(k)}(x)$ в силу дифференциального уравнения (1)

$$\begin{array}{l} y'(x) = f(x, y(x)), \\ y''(x) = \frac{d}{dx}f(x, y(x)) = f'_x + f'_y f, \\ y'''(x) = \frac{d}{dx}\left(f'_x + f'_y f\right) = f''_{xx} + f'_x f'_y + \\ + \left(\left(f'_y\right)^2 + 2f''_{xy}\right) f + f''_{yy}f^2, \dots, \end{array}$$

получаем последовательность приближений

$$y_{n+1} = y_n + h\varphi_p(x_n, y_n, h),\ n = 0, 1, 2, \dots. \tag{3}$$

Отсюда, в частности, при $p = 1$ получаем схему

$$\begin{cases} y_{n+1} = y_n + hf(x_n, y_n) \\ y_0 = y^0,\ n = \overline{0, N-1},\ x_n \in \omega_N, \end{cases} \tag{4}$$

(*метод Эйлера*), а при $p = 2$ - схему

$$\begin{cases} y_{n+1} = y_n + h[f(x_n, y_n) + \frac{h}{2}(f_x(x_n, y_n) + f_y(x_n, y_n)f(x_n, y_n)] \\ y_0 = y^0,\ n = \overline{0, N-1},\ x_n \in \omega_N. \end{cases} \tag{5}$$

Алгоритмы типа (3) не требуют вычисления дополнительных начальных условий и позволяют легко менять шаг интегрирования. Применимость данных методов ограничена теми задачами, для которых легко вычисляются частные производные высоких порядков функции $f(x, y)$.

1.3 Методы Рунге-Кутты

Идея этих методов заключается в построении функций φ, которые не используют явно производных функции $f(x,y)$ и в то же время "наилучшим образом" приближают соответствующие отрезки φ_p ряда Тейлора $\triangle$.

Продемонстрируем процесс "подгонки" рядов Тейлора следующим образом. Положим

$$\varphi(x,y,h) \equiv c_1 f(x,y) + c_2 f(x + ha_2, y + b_{21} h f(x,y)), \tag{6}$$

где c_1, c_2, a_2, b_{21} - постоянные, подлежащие определению. Разлагая φ по степеням h до членов порядка h^2, получим

$$\varphi(x,y,h) = (c_1 + c_2) f(x,y) + h c_2 [a_2 f_x(x,y) + b_{21} f_y(x,y) f(x,y)] + O(h^2). \tag{7}$$

С другой стороны,

$$\triangle(x,y,h) = f(x,y) + \frac{1}{2} h [f_x(x,y) + f_y(x,y) f(x,y)] + O(h^2). \tag{8}$$

Отсюда видно, что взяв

$$\begin{cases} c_1 + c_2 = 1 \\ c_2 a_2 = \frac{1}{2} \\ c_2 b_{21} = \frac{1}{2}, \end{cases} \tag{9}$$

получаем в (6) функцию, для которой

$$\varphi(x,y,h) - \triangle(x,y,h) = O(h^2).$$

Алгебраическая система (9) имеет множество решений вида

$$c_1 = 1 - \alpha,\ c_2 = \alpha,\ a_2 = b_{21} = \frac{1}{2\alpha},$$

где α - свободный параметр.

При $\alpha = \frac{1}{2}$, в частности, получаем двукратный[1] *метод Рунге-Кутты-метод Хойна:*

$$\begin{cases} y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_n + h f(x_n, y_n))], \\ y_0 = y^0,\ n = \overline{0, N-1},\ x_n \in \omega_N, \end{cases} \tag{10}$$

требующий всего два вычисления функции $f(x,y)$ на каждом шаге.

[1]Под кратностью метода в данном случае понимается количество вычислений функции $f(x,y)$.

Процесс "подгонки" рядов Тейлора можно продолжить, строя функции $\varphi(x, y, h)$, использующие все большие отрезки ряда Тейлора. Подобным образом получают *m-кратные явные методы Рунге-Кутты:*

$$\begin{cases} y_{n+1} = y_n + h\varphi(x_n, y_n, h) \\ y_0 = y^0,\ n = \overline{0, N-1},\ x_n \in \omega_N, \end{cases} \tag{11}$$

$$\varphi(x, y, h) \equiv \sum_{r=1}^{m} c_r k_r,$$

$$k_1 \equiv f(x, y),\ k_r \equiv f(x + ha_r, y + h\sum_{s=1}^{r-1} b_{rs}k_s),\ r = \overline{2, m},$$

требующие m вычислений функции $f(x, y)$ на каждом шаге.

Наиболее известным из них является четырехкратный метод:

$$\begin{cases} y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ y_0 = y^0,\ n = \overline{0, N-1},\ x_n \in \omega_N, \end{cases}$$

$$\begin{aligned} k_1 &\equiv f(x_n, y_n), \\ k_2 &\equiv f(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}hk_1), \\ k_3 &\equiv f(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}hk_2), \\ k_4 &\equiv f(x_n + h, y_n + hk_3). \end{aligned}$$

Алгоритмы Рунге-Кутты отлично приспособлены для практических вычислений: они не требуют вычисления дополнительных начальных данных и легко позволяют менять шаг h.

Усложняя формулы (11), можно получить и более широкий класс методов, т.н. *m−кратные неявные методы Рунге-Кутты*:

$$\begin{cases} y_{n+1} = y_n + h\varphi(x_n, y_n, h) \\ y_0 = y^0,\ n = \overline{0, N-1},\ x_n \in \omega_N, \end{cases} \tag{12}$$

$$\varphi(x, y, h) \equiv \sum_{r=1}^{m} c_r k_r,$$

$$k_1 \equiv f(x, y),\ k_r \equiv f(x + ha_r, y + h\sum_{s=1}^{m} b_{rs}k_s),\ r = \overline{2, m}.$$

Формулы (12) являются, вообще говоря, более точными, чем (11). Однако более высокая точность достигается ценой значительного усложнения вычислительного алгоритма, что сильно ограничивает сферу применимости таких методов. Можно, впрочем, понизить степень неявности,

положив $b_{rs} = 0$ при $s > r$. В качестве примера приведем полуявный трехкратный *метод Батчера*

$$\begin{cases} y_{n+1} = y_n + \frac{h}{6}(k_1 + 4k_2 + k_3), \\ y_0 = y^0,\ n = \overline{0, N-1},\ x_n \in \omega_N, \end{cases} \tag{13}$$

$$\begin{aligned} k_1 &\equiv f(x_n, y_n), \\ k_2 &\equiv f(x_n + \tfrac{h}{2}, y_n + \tfrac{h}{4}(k_1 + k_2)), \\ k_3 &\equiv f(x_n + h, y_n + hk_2). \end{aligned}$$

Одной из проблем, связанных с реализацией методов Рунге-Кутты, является выбор шага для достижения заданной точности. Решается она либо применением *правила Рунге*[2], что значительно увеличивает объем вычислений, либо путем использования специальных формул для вычисления погрешности, содержащих только величины $\{k_r\}$, (*метод Мерсона*). Например, для явного пятикратного метода

$$\begin{cases} y_{n+1} = y_n + \frac{h}{2}(k_1 + 4k_4 + k_5) \\ y_0 = y^0,\ n = \overline{0, N-1},\ x_n \in \omega_N, \end{cases}$$

$$\begin{aligned} k_1 &\equiv \tfrac{1}{3} f(x_n, y_n), \\ k_2 &\equiv \tfrac{1}{3} f(x_n + \tfrac{h}{3}, y_n + hk_1), \\ k_3 &\equiv \tfrac{1}{3} f(x_n + \tfrac{h}{3}, y_n + \tfrac{h}{2}k_1 + \tfrac{h}{2}k_2), \\ k_4 &\equiv \tfrac{1}{3} f(x_n + \tfrac{h}{2}, y_n + \tfrac{3}{8}hk_1 + \tfrac{9}{8}hk_3), \\ k_5 &\equiv \tfrac{1}{3} f(x_n + h, y_n + \tfrac{3}{2}hk_1 - \tfrac{9}{2}hk_3 + 6hk_4), \end{aligned}$$

погрешность метода Мерсона вычисляется по формуле

$$\varepsilon = \frac{h}{5}(k_1 - \frac{9}{2}k_3 + 4k_4 - \frac{1}{2}k_5). \tag{14}$$

Критерий выбора шага интегрирования состоит в следующем: если правая часть (14) превышает предписанную погрешность более чем в 5 раз, то шаг h уменьшается в 2 раза и вычисления повторяются до достижения предписанной погрешности, а если правая часть (14) меньше, чем $\frac{5}{32}$ предписанной погрешности, то шаг может быть удвоен и вычисления продолжаются с удвоенным шагом.

1.4 Многошаговые методы

В изложенных выше методах, после нахождения приближенного решения y_{n+1} на $n+1$-ом шаге, приближенное решение y_n уже не используется в дальнейших расчетах. Идея многошаговых методов состоит в том,

[2]Сравниваются решения, полученные с шагами h и $h/2$. В случае их совпадения с указанной погрешностью решение считается удовлетворительным.

чтобы явно использовать ранее вычисленные значения $y_n, y_{n-1}, \ldots$. Реализовать ее можно, например, следующим способом. Задачу Коши (1) приводят к эквивалентному интегральному уравнению

$$\begin{gathered} y(x+\zeta) - y(x) = \int\limits_{x}^{x+\zeta} f(t, y(t))dt, \\ y(a) = y^0, \end{gathered} \tag{15}$$

где $x, x+\zeta$ - произвольные точки интервала $[a, b]$.

Функцию $F(x) \equiv f(x, y(x))$ заменяют интерполяционным многочленом, принимающим значения $F_n = f(x_n, y_n)$ на множестве точек сетки x_n, в которых должны быть вычислены приближенные значения y_n. Пусть $x_n, x_{n-1}, \ldots, x_{n-k}$ - узлы интерполяции, тогда интерполяционный многочлен можно записать, например, в форме Лагранжа,

$$L_n^k(x) = \sum_{i=0}^{k} p_{ki}(x) F_{n-i}. \tag{16}$$

Здесь $p_{ki}(x)$ многочлены Лагранжа.

Подставляя (16) в уравнение (15) для $x = x_n, \zeta = h$ и выполняя интегрирование, получаем *схему Адамса-Бешфорта*

$$\begin{cases} y_{n+1} = y_n + h \sum_{i=0}^{k} \beta_{ki} F_{n-i} \\ y_0 = y^0, \ n = \overline{0, N-1}, \ x_n \in \omega_N, \end{cases} \tag{17}$$

$$\beta_{ki} \equiv \frac{1}{h} \int\limits_{x_n}^{x_n+h} p_{ki}(t)dt.$$

При $x = x_{n-1}, \zeta = h$ получим *схему Адамса-Мултона*

$$\begin{cases} y_n = y_{n-1} + h \sum_{i=0}^{k} \beta_{ki} F_{n-i} \\ y_0 = y^0, \ n = \overline{0, N-1}, \ x_n \in \omega_N, \end{cases} \tag{18}$$

$$\beta_{ki} \equiv \frac{1}{h} \int\limits_{x_n-h}^{x_n} p_{ki}(t)dt.$$

Общая форма записи таких методов:

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j F_{n+j}, \ n = 0, 1, \ldots, \tag{19}$$

где α_j, β_j - постоянные, $\alpha_n \neq 0, \ |\alpha_0| + |\beta_0| \neq 0$. Уравнение (19) - линейное соотношение между y_{n+j} и F_{n+j}. Поэтому соответствующий формуле (19) метод называют *линейным многошаговым методом* (LLM - linear

multistep method). Для применения схемы (19) требуется заранее знать k начальных значений $y_0, y_1, \ldots, y_{k-1}$. Эти значения получают каким-либо одношаговым методом. Далее процесс может следовать по одному из двух возможных путей. Если $\beta_k = 0$, то y_{n+k} легко вычисляются из формул (19), и метод называют *явным многошаговым*. В противном случае правая часть $F_{n+k} \equiv f(x_{n+k}, y_{n+k})$ содержит искомую величину y_{n+k}, и для ее определения приходится решать (нелинейное в общем случае) уравнение. Метод при этом называют *неявным многошаговым*. Формулы Адамса (17), (18) являются примерами соответственно явных и неявных методов. Явные многошаговые методы на первый взгляд кажутся самыми простыми для проведения вычислений. Но на практике такие методы используются редко, т.к. они обладают меньшей устойчивостью к малым возмущениям значений y_n (ошибки округления). Оказывается также, что в явных методах шаг h должен быть значительно меньше для достижения заданной точности, чем в неявных, и в случае т.н. "жестких" дифференциальных уравнений явные методы нередко приводят к неверным результатам.

Отметив, что неявные методы предпочтительнее явных, покажем как они реализуются на практике. Из (19) имеем

$$y_{n+k} = h\frac{\beta_k}{\alpha_k} f(x_{n+k}, y_{n+k}) + g_n, \ \beta_k \neq 0, \tag{20}$$

где g_n содержит уже известные величины $y_{n+j}, F_{n+j}, \ j = \overline{0, k-1}$. Оказывается, что если

$$hL < \left|\frac{\alpha_k}{\beta_k}\right|, \tag{21}$$

где L-постоянная Липшица, определенная в п. 1.1, то единственное решение нелинейного уравнения (20) можно получить с помощью итерационного процесса

$$y_{n+k}^{\nu+1} = h\frac{\beta_k}{\alpha_k} f(x_{n+k}, y_{n+k}^{\nu}) + g_n, \ \nu = 0, 1, \ldots. \tag{22}$$

Скорость сходимости итерационной последовательности тем выше, чем меньше величина $h|\beta_k/\alpha_k|L$. Величина y_{n+k}^0 выбирается произвольно. Желательно, конечно, выбирать y_{n+k}^0 вблизи искомой величины. Надлежащий выбор достигается с помощью явного метода. Явный метод в этом случае называют "*предсказывающим*", а неявный метод (20) - "*исправляющим*". В литературе весь процесс принято называть методом

"*предиктора-корректора*". [3]

Существует два способа реализации методов предиктора-корректора. В первом случае итерации (22) проводят до тех пор, пока они не будут совпадать с заданной точностью. Во втором случае корректирующие итерации применяются фиксированное, например t , число раз. Полученное таким образом значение y_{n+k}^t принимается за приближение к точному значению $y(x_{n+k})$.

В качестве примера рассмотрим методы Адамса при $k = 1$. Здесь формулы для явного метода имеют вид

$$y_{n+1} = y_n + \frac{h}{2}(3f(x_n, y_n) - f_(x_{n-1}, y_{n-1}),$$

а формулы для неявного метода

$$y_{n+1} = y_n + \frac{h}{12}(5f(x_{n+1}, y_{n+1}) + 8f(x_n, y_n) - f(x_{n-1}, y_{n-1})).$$

Для $t = 1$ формулы метода предиктора-корректора можно записать в виде

$$\begin{cases} y_{n+1}^0 = y_n + \frac{h}{2}(3f(x_n, y_n) - f_(x_{n-1}, y_{n-1}), \\ y_{n+1} = y_n + \frac{h}{12}(5f(x_{n+1}, y_{n+1}^0) + 8f(x_n, y_n) - f(x_{n-1}, y_{n-1})). \end{cases}$$

Главной трудностью при использовании многошаговых методов является изменение шага интегрирования. Преимущество перед одношаговыми - меньшее число вычислений правой части дифференциального уравнения для достижения одинаковой точности.

1.5 Системы уравнений первого порядка в нормальной форме и уравнения высокого порядка

Коротко продемонстрируем, как изложенные выше методы можно применять к системам первого порядка и уравнениям высших порядков. Для системы из s уравнений первого порядка

$$\begin{cases} y' = f(x, y), \ x \in (a, b) \\ y(a) = y^0, \end{cases} \tag{23}$$

где

$$y\,(x) \equiv [y^1(x), y^2(x), \ldots, y^s(x)],$$
$$f(x, y) \equiv [f^1(x, y), f^2(x, y), \ldots, f^s(x, y)],$$

[3] Отметим, что идея предиктора-корректора аналогичным образом применима и для одношаговых методов[2].

метод разложения в ряд Тейлора можно применить покомпонентно. А именно, разложим каждую из функций $y^i(x)$, $i = \overline{1,s}$ в ряд Тейлора в окрестности точки x_n

$$y^i(x_n + h) = y^i(x_n) + h\frac{dy^i}{dx}(x_n) + \frac{h^2}{2}\frac{d^2y^i}{dx^2}(x_n) + \dots.$$

Обрывая этот ряд, получаем соответствующие обобщения методов п. 1.2. Например, отбрасывая члены порядка h^3 и выше, получаем в покомпонентной форме

$$\begin{gathered} y^i_{n+1} = y^i_n + h\left(f^i(x_n, y^1_n, y^2_n, \dots, y^s_n) + \right. \\ \left. + \frac{h}{2}\left(\frac{\partial f^i}{\partial x} + \sum\limits_{j=1}^{s}\frac{\partial f^i}{\partial y^j}f^j(x_n, y^1_n, y^2_n, \dots, y^s_n)\right)\right), \\ i = \overline{1,s}. \end{gathered} \tag{24}$$

В векторной записи этот метод тождественен по форме методу (5) для скалярного уравнения:

$$y_{n+1} = y_n + h\left(f(x_n, y_n) + \frac{h}{2}f'(x_n, y_n)\right), \tag{25}$$

где теперь

$$f'(x,y) \equiv \frac{\partial f}{\partial x}(x,y) + J(x,y)f(x,y),$$

$J(x,y) \equiv (\partial f^i/\partial y^j)$ - матрица Якоби функции f.

Методы Рунге-Кутты, рассмотренные выше, также можно применять к системам, заменив скаляры $y, f, \varphi, k_j, j = \overline{1,m}$, соответствующими s-мерными векторами.

Идея построения многошаговых методов состоит в приближенном вычислении интегралов в системе интегральных уравнений

$$\begin{gathered} y^i(x+\zeta) = y^i(x) + \int\limits_x^{x+\zeta} f^i(t, y^1(t), y^2(t), \dots, y^s(t))dt, \\ y(a) = y^0,\ i = \overline{1,s}, \end{gathered}$$

эквивалентных задаче Коши (23). Для получения разностных схем вида (15)-(19), достаточно повторить рассуждения предыдущего раздела, считая y_{n+j}, F_{n+j}, $j = \overline{0,k}$ s-мерными векторами.

Задачу Коши для уравнения p-го порядка, разрешенного относительно старшей производной

$$\begin{cases} y^{(p)}(x) = f(x, y'(x), y''(x), \dots, y^{(p-1)}),\ x \in [a,b], \\ y(a) = \beta_1,\ y'(a) = \beta_2, \dots,\ y^{(p-1)}(a) = \beta_p, \end{cases} \tag{26}$$

сводят введением новых функций

$$y^i(x) \equiv y^{(i-1)}(x),\ i = \overline{1,p},$$

к эквивалентной задаче Коши для системы первого порядка

$$\begin{cases} \frac{d}{dx}y^i(x) = y^{i+1}(x),\ i = \overline{1,p-1} \\ \frac{d}{dx}y^p(x) = f(x, y^1(x), y^2(x), \ldots, y^p(x)) \\ y^1(a) = \beta_1,\ y^2(a) = \beta_2, \ldots,\ y^p(a) = \beta_p. \end{cases}$$

Для численного решения этой системы можно применять методы, изложенные в настоящем пункте выше.

Методы Рунге-Кутты и многошаговые методы также допускают обобщения на случай задачи (26) (см. [4], [5]).

2 Устойчивость разностных схем

Рассмотренные выше алгоритмы численного решения задачи Коши, являются алгоритмами рекуррентного типа, и поэтому допускаемые в процессе вычислений ошибки различного рода могут накапливаться. Не следует также забывать, что входные данные исходной задачи (правые части и начальные условия) обычно задаются лишь с определенной точностью. Естественно требовать от алгоритма, с помощью которого мы решаем задачу Коши, чтобы ошибки, допущенные во входных данных и ошибки, связанные с округлением в процессе вычислений, не приводили к значительным искажениям решения. Алгоритмы, которые в процессе вычислений усиливают погрешности называются *неустойчивыми* и, как правило, не используются на практике.

Прежде, чем переходить к изложению вопроса устойчивости вычислительных алгоритмов решения задачи Коши (23), коротко остановимся на вопросе устойчивости решения самой задачи Коши по начальным данным и правым частям.

Пусть $u(x), v(x)$ - решения задачи Коши (23) с начальными данными $u(0) = u_0,\ v(0) = v_0$. Их разность $z(x)$ удовлетворяет системе линейных уравнений

$$\frac{dz}{dx} = \Lambda(x)\, z,$$

где $\Lambda(x)$ - матрица Якоби вектор-функции $f(x, y)$, вычисленная в некоторых средних точках $\overline{u}_k(x),\ k = \overline{1,s}$:

$$f^k(x, u(x)) - f^k(x, v(x)) \equiv \sum_{i=1}^{s} \frac{\partial f^k}{\partial x_i}(x, \overline{u}_k(x)) \left(u^i(x) - v^i(x)\right) \equiv \\ \equiv \sum_{i=1}^{s} \Lambda_{ki}(x) \left(u^i(x) - v^i(x)\right).$$

Поэтому задачу

$$\frac{dz}{dx} - \Lambda(x)\, z = 0,\ z(0) = z_0, \tag{27}$$

можно рассматривать как модельную при исследовании свойств устойчивости по начальным данным решения задачи (23).

Вопросам устойчивости решений линейных систем посвящена обширная литература (см., например,[1]). В частности, в случае постоянной матрицы Λ для асимптотической устойчивости решения задачи (27) по возмущению начальных данных достаточно, чтобы матрица была самосопряженной, отрицательно определенной[4]. При этом для решения задачи выполняется оценка

$$\|z(x)\| \le e^{\lambda_1 x}\|z(0)\|,\ x > 0,$$

где $\lambda_1 < 0$ - наименьшее по модулю собственное число матрицы Λ.

Аналогичные подходы используются и при изучении свойств устойчивости вычислительных алгоритмов решения задачи Коши. Прежде чем давать более или менее строгие определения устойчивости вычислительных алгоритмов решения задачи Коши рассмотрим несколько примеров [3] для модельного уравнения

$$\begin{cases} u' + \alpha u = 0, x > 0 \\ u(0) = u_0. \end{cases} \tag{28}$$

Точным решением уравнения (28) является функция $u(x) = u_0 e^{-\alpha x}$. Это решение при $\alpha > 0$ убывает с ростом $x : |u(x)| \le |u_0|\, e^{-\alpha x} \to 0$ и непрерывно зависит от начального значения u_0. Естественно ожидать выполнения этих условий и для приближенного решения задачи.

Пример 1. Устойчивая схема.

Задачу (28) на равномерной сетке $\omega_h = \{x_i = ih, i = 0, 1, \ldots\}$ аппроксимируем неявной схемой Эйлера

$$y_i = y_{i-1} - h\alpha y_i,\ y_0 = u_0,\ i = 1, 2, \ldots.$$

Переписывая это уравнение в виде

$$y_i = \frac{1}{1+\alpha h} y_{i-1},\ y_0 = u_0,\ i = 1, 2, \ldots.$$

получаем

$$y_i = \left(\frac{1}{1+\alpha h}\right)^i y_0,\ i = 1, 2, \ldots.$$

[4]Т.е. симметричной и удовлетворяющей условию $(\Lambda y, y) < 0\ \forall y \neq 0$. Заметим, что любая отрицательно определенная матрица обратима.

Рассмотрим фиксированную точку $\overline{x}$ сетки ω_h и выберем такую последовательность шагов h, чтобы точка $\overline{x}$ все время оставалась узловой точкой, т.е. $\overline{x} = i_0 h$. Тогда при измельчении сетки ($h \to 0$) номер i_0, соответствующий выбранной точке $\overline{x}$, неограниченно возрастает. Значение приближенного решения в этой точке

$$y_{i_0} = \left(\frac{1}{1+\alpha h}\right)^{i_0} y_0.$$

Так как $1/|1+\alpha h| < 1$, то

$$|y_{i_0}| \le |\frac{1}{1+\alpha h}|^{i_0} |y_0| < |y(0)|.$$

Из последнего неравенства следует, что решение разностной задачи непрерывно зависит от начальных данных и, следовательно, ошибки, допущенные в начальных данных и ошибки округления, которые появляются при вычислениях, не нарастают. В таких случаях говорят, что алгоритм вычислений *устойчив*.

Пример 2. Неустойчивая схема.

Для решения задачи (28) на равномерной сетке ω_h рассмотрим схему

$$\begin{cases} \sigma \frac{y_i - y_{i-1}}{h} + (1-\sigma)\frac{y_{i+1}-y_i}{h} + \alpha y_i = 0 \\ y_0 = u_0,\ y_1 = \overline{u}_0,\ i = 1, 2, \dots, \end{cases} \tag{29}$$

где $\sigma > 1$ - числовой параметр. Решение задачи (29) имеет вид

$$y_i = A s_1^i + B s_2^i, \tag{30}$$

где

$$s_{1,2} \equiv \frac{2\sigma - 1 + \alpha h \pm \sqrt{1 + 2(\sigma-1)\alpha h + \alpha^2 h^2}}{2(\sigma-1)},$$

$$A \equiv \frac{\overline{u}_0 - s_2 u_0}{s_1 - s_2},\ B \equiv \frac{s_1 u_0 - \overline{u}_0}{s_1 - s_2}.$$

Так как $\sigma > \sigma - 1 > 0$, то $s_1 s_2 > 1$. Кроме того, $s_2 < 1,\ s_1 > 1$ при любых значениях αh. Из (30) видно, что $y_i \to \infty$ при $i \to \infty$, если $A \neq 0$.

Таким образом, при фиксированном h схема (29) приводит к нарастанию решения с ростом $x_i = ih$. Сгущение сетки (т.е. уменьшение h) приводит к нарастанию решения в фиксированной точке $\overline{x} = i_0 h$, так как $i_0 = \overline{x}/h$ растет с уменьшением h. Малое изменение начальных данных $u_0, \overline{u}_0$ приводит при $h \to 0$ к неограниченному возрастанию ошибки в любой заданной точке.

Сравнительно просто дать определение устойчивости и привести критерии устойчивости для двухслойных разностных схем для модельной задачи

$$\begin{cases} \frac{du}{dx} + A(x)u = \varphi(x), \ x > 0 \\ u(0) = u_0. \end{cases} \tag{31}$$

Здесь $A\,(x)$ - квадратная матрица порядка $s \times s$ с элементами, зависящими от x, $u(x), \varphi(x)$ - вектор-функции размерности s.

Под *двухслойной схемой* понимают разностное уравнение, связывающее значение вектора $y(x)$ для двух слоев значений аргументов $x = x_n$ и $x = x_{n+1}$. В канонической форме это соотношение задают в виде:

$$\begin{cases} B\frac{y_{n+1}-y_n}{h} + Ay_n = \varphi_n \\ n = 0, 1, 2, \ldots, \ y_0 = u_0, \end{cases} \tag{32}$$

где B, A - квадратные матрицы порядка $s \times s$, y_n, φ_n - векторы размерности s. Если $B = E$ - единичной матрице, то каноническую схему (32) называют *явной*, в противном случае - *неявной*.

Конкретные часто используемые схемы:

- *чисто неявная схема* $(B = E + hA)$

$$\begin{cases} \frac{y_{n+1}-y_n}{h} + Ay_{n+1} = \varphi_n \\ n = 0, 1, 2, \ldots, \ y_0 = u_0; \end{cases} \tag{33}$$

- *симметричная схема* $\left(B = E + \frac{1}{2}hA\right)$

$$\begin{cases} \frac{y_{n+1}-y_n}{h} + \frac{1}{2}A(y_n + y_{n+1}) = \varphi_n \\ n = 0, 1, 2, \ldots, \ y_0 = u_0; \end{cases} \tag{34}$$

- *схема с весами* $(B = E + \sigma hA)$

$$\begin{cases} \frac{y_{n+1}-y_n}{h} + A((1-\sigma)y_n + \sigma y_{n+1}) = \varphi_n \\ n = 0, 1, 2, \ldots, \ y_0 = u_0. \end{cases} \tag{35}$$

Пусть $\|.\|_{(1)}$,$\|.\|_{(2)}$ - некоторые нормы в R_s.

Определение. Каноническую схему (32) назовем *устойчивой по начальным данным и правой части*, если существуют положительные постоянные M_1, M_2, не зависящие от h, n, y_0, φ_n,такие, что для решения y_n задачи (32) имеет место оценка

$$\|y_n\|_{(1)} \leq M_1\|y_0\|_{(1)} + M_2 \max_{0\leq k\leq n-1} \|\varphi_k\|_{(2)}. \tag{36}$$

Удобным средством исследования устойчивости канонической схемы оказывается следующее понятие равномерной устойчивости.

Определение. Каноническую схему (32) назовем *равномерно устойчивой по начальным данным*, если существует положительная постоянная ρ, не зависящая от h, n, y_0, такая, что для решения однородной задачи (32)

$$\begin{cases} B\frac{y_{n+1}-y_n}{h} + Ay_n = 0 \\ n = 0, 1, 2, \ldots, \ y_0 = u_0, \end{cases} \tag{37}$$

имеет место оценка

$$\|y_{n+1}\|_{(1)} \leq \rho \|y_n\|_{(1)}, \ n = 0, 1, \ldots.$$

Переписав уравнение (37) в операторной форме

$$y_{n+1} = Sy_n,$$

где $S \equiv E - hB^{-1}A$ - *оператор перехода,* нетрудно показать, что введенное выше условие равномерной устойчивости эквивалентно ограниченности нормы оператора перехода:

$$\|S\| \leq \rho. \tag{38}$$

С другой стороны, можно показать, что из равномерной устойчивости схемы по начальным условиям вытекает ее устойчивость по начальным данным и правой части с постоянными M_1, M_2, вычисляемыми через постоянную ρ, и нормой $\|\varphi\|_{(2)} \equiv \left\|B^{-1}\varphi\right\|_{(1)}$.

Наконец, условие ограниченности нормы оператора перехода (38) легко выразить в терминах исходных операторов A, B.

Пусть D - самосопряженная положительная матрица. Вводя в R_s энергетическую норму

$$\|y\|_D \equiv \sqrt{(Dy, y)},$$

где $(\cdot, \cdot)$ - скалярное произведение, превратим это линейное пространство в нормированное пространство H_D.

Теорема. Если A - самосопряженная положительная матрица и существует обратная матрица B^{-1}, то для равномерной устойчивости (с постоянной $\rho = 1$) канонической схемы (32) по начальным данным в H_A:

$$\|y_n\|_A \leq \|y_0\|_A,$$

необходимо и достаточно, чтобы выполнялось неравенство

$$(By, y) - \frac{h}{2}(Ay, y) \geq 0 \ \ \forall y \in R_s,$$

или

$$B \geq \frac{h}{2}A. \tag{39}$$

Таким образом, для исследования устойчивости конкретной двухслойной схемы достаточно записать ее в канонической форме (32), определить соответствующие операторы A, B и проверить для них выполнение условий данной теоремы.

В частности, для чисто явной схемы ($B = E$) условие устойчивости (39) можно записать в виде

$$h \leq \frac{2}{\|A\|}.$$

Схема с весами (35) устойчива для любых $h > 0$ (*безусловно устойчива*), если $\sigma \geq \frac{1}{2}$, и *условно устойчива* при $h(\frac{1}{2} - \sigma)\|A\| \leq 1$, если $\sigma < \frac{1}{2}$. Поэтому схемы (33) и (34) являются безусловно устойчивыми.

2.1 Погрешность аппроксимации и сходимость приближенного решения

Пусть $u(x)$ - точное решение задачи (31), y_n - решение, полученное с помощью канонической схемы (32). Величину $z_n \equiv y_n - u(x_n)$ назовем *погрешностью решения*. Подставляя y_n в (32) получим уравнения для погрешности решения

$$\begin{cases} B\frac{z_{n+1}-z_n}{h} + Az_n = \psi_n \\ n = 0, 1, 2, ..., \ z_0 = 0, \end{cases} \tag{40}$$

где

$$\psi_n \equiv \varphi_n - Au(x_n) - B\frac{u\,(x_{n+1}) - u(x_n)}{h}$$

- величина, которую называют *погрешностью аппроксимации* для схемы (32) на решении $u(x)$ исходной задачи (31) (*невязкой*). При этом говорят, что схема (32) имеет $m-$*ый порядок аппроксимации* на решении задачи (31), если для невязки ψ_n выполняется оценка

$$\|\psi_n\|_{(2)} = O(h^m).$$

Далее, говорят, что схема (32) имеет m-*ый порядок точности* или *сходится со скоростью* $O(h^m)$, если

$$\|z_n\|_{(1)} = O(h^m),$$

т.е.

$$\|z_n\|_{(1)} \le Mh^m,$$

где $M > 0$ - постоянная, не зависящая от h, n.

Найдем условия аппроксимации и сходимости приближенного решения полученного с помощью канонической схемы (32) задачи (31). Предполагая, что решение задачи (31) дважды дифференцируемо, имеем

$$u(x_{n+1}) = \left(u + \frac{h}{2}\frac{du}{dx} + \frac{h^2}{8}\frac{d^2u}{dx^2}\right)|_{x=x_{n+\frac{1}{2}}} + O(h^3), \tag{41}$$

$$u(x_n) = \left(u - \frac{h}{2}\frac{du}{dx} + \frac{h^2}{8}\frac{d^2u}{dx^2}\right)|_{x=x_{n+\frac{1}{2}}} + O(h^3), \tag{42}$$

где $x_{n+\frac{1}{2}} \equiv x_n + h/2$.

Вычитая из равенства (41) равенство (42), имеем

$$\frac{u(x_{n+1}) - u(x_n)}{h} = \frac{du}{dx}|_{x=x_{n+\frac{1}{2}}} + O(h^2). \tag{43}$$

Используя (42),(43), для погрешности аппроксимации будем иметь равенство

$$\psi_n = \varphi_n - \left(Au + B\frac{du}{dx}\right)|_{x=x_{n+\frac{1}{2}}} + \frac{h}{2}A\frac{du}{dx}|_{x=x_{n+\frac{1}{2}}} + O(h^2) =$$

$$= \varphi_n - \varphi(x_{n+\frac{1}{2}}) + \left(\varphi - Au - \frac{du}{dx}\right)|_{x=x_{n+\frac{1}{2}}} + \left(E - B + \frac{h}{2}A\right)\frac{du}{dx}|_{x=x_{n+\frac{1}{2}}} + O(h^2) =$$

$$= \varphi_n - \varphi(x_{n+\frac{1}{2}}) + \left(E - B + \frac{h}{2}A\right)\frac{du}{dx}|_{x=x_{n+\frac{1}{2}}} + O(h^2).$$

Отсюда видно, что при выполнении условий

$$\|\varphi_n - \varphi(x_{n+\frac{1}{2}})\|_{(2)} = O(h^m),\ m = \overline{1,2},$$

$$\|\left(E - B + \frac{h}{2}A\right)\frac{du}{dx}\|_{(2)} = O(h^m),\ m = \overline{1,2},$$

схема (32) имеет m-ый порядок аппроксимации.

В частности, для чисто явной схемы ($B = E$)

$$\|\left(E - B + \frac{h}{2}A\right)\frac{du}{dx}\|_{(2)} = O(h).$$

Поэтому, если

$$\|\varphi_n - \varphi(x_{n+\frac{1}{2}})\|_{(2)} = O(h),$$

то чисто явная схема имеет 1-ый порядок аппроксимации.

Для симметрической схемы $\left(B = E + \frac{h}{2}A\right)$

$$\| \left(E - B + \frac{h}{2}A\right) \frac{du}{dx} \|_{(2)} = 0$$

и поэтому при

$$\|\varphi_n - \varphi(x_{n+\frac{1}{2}})\|_{(2)} = O(h^2)$$

порядок аппроксимации - 2-ой.

Рассмотрим далее вопрос о сходимости приближенного решения, полученного с помощью канонической схемы (32) к точному решению исходной задачи (31). Предполагая, что схема (32) устойчива из (36) имеем

$$\|z_n\|_{(1)} \leq M \max_{0 \leq k \leq n-1} \|\psi_k\|_{(2)}.$$

Считая, что схема (32) имеет некоторый порядок аппроксимации, получаем, что $\|z_n\|_{(1)} \to 0$ при $h \to 0$.

Из сказанного следует, что исследование сходимости решения и порядка его точности сводится к изучению порядка аппроксимации и устойчивости разностной схемы.

3 Практическая часть

3.1 Вопросы для самопроверки

1. Метод разложения в ряд Тейлора построения разностных схем для задачи Коши.
2. Явные методы Рунге-Кутты.
3. Неявные методы Рунге-Кутты.
4. Многошаговые методы Адамса.
5. Метод Эйлера для решения систем обыкновенных дифференциальных уравнений.
6. Каноническая двухслойная схема для задачи Коши; схема с весами.
7. Понятие устойчивости разностной схемы по начальным данным и правым частям; равномерная устойчивость по начальным данным.
8. Критерии устойчивости канонической схемы.
9. Порядок аппроксимации и скорость сходимости разностной схемы.

3.2 Варианты заданий лабораторной работы

В лабораторной работе требуется найти решение задачи Коши. Варианты заданий приведены в таблице 1. В каждом задании необходимо:

a) для заданного уравнения провести теоретическое исследование устойчивости и порядка аппроксимации заданного метода численного решения;

b) написать программу для решения заданной задачи Коши заданным методом;

c) решить заданную задачу Коши с помощью подпрограммы RKF45, приведенной в приложении и найти точное решение;

d) провести сравнение, полученных решений в табличной и графической форме.

Таблица 1.

Уравнение	Начальные условия	Область решения	Точность	Метод решения
$y'' + y = sin(x)$	$y(0) = 1, y'(0) = 0$	$[0, 2\pi]$	0.01	Рунге-Кутты 2-го порядка
$y'' + y = cos(x)$	$y(0) = 1, y'(0) = 0$	$[0, 2\pi]$	0.01	Хойна
$y'' - y = exp(x)$	$y(0) = 1, y'(0) = 0$	$[0, 2]$	0.01	Батчера
$y'' - y = exp(-x)$	$y(0) = 1, y'(0) = 0$	$[0, 2]$	0.01	Эйлера
$y'' - 2y' + y = exp(x)$	$y(0) = 1, y'(0) = 0$	$[0, 2]$	0.01	Чисто неявная схема
$y'' + 2y' + y = exp(-x)$	$y(0) = 1, y'(0) = 0$	$[0, 2]$	0.01	Тейлора 2-го порядка
$y'' + 2y' + y = xexp(-x)$	$y(0) = 1, y'(0) = 0$	$[0, 2]$	0.01	Рунге-Кутты 4-го порядка
$y'' + 2y' + 2y = xsin(x)$	$y(0) = 1, y'(0) = 0$	$[0, 2\pi]$	0.01	Симметричная схема
$y'' + 2y' + 2y = xcos(x)$	$y(0) = 1, y'(0) = 0$	$[0, 2\pi]$	0.01	Мерсона

3.3 Требования к программам

Подпрограмму, реализующую алгоритм решения задачи Коши, следует сделать независимой от конкретного вида уравнения и начальных данных. Этого можно достичь выносом вычисления правых частей в отдельную подпрограмму. Так, например, как это сделано в подпрограмме RKF45, представленной в приложении.

Список литературы

[1] Понтрягин Л.С. Обыкновенные дифференциальные уравнения. М.:Наука. 1970. -332 с.

[2] Самарский А.А. Введение в численные методы. М.:Наука. 1982. -272 с.

[3] Самарский А.А., Гулин А.В. Численные методы. М.:Наука. 1989. -430 с.

[4] Современные численные методы решения обыкновенных дифференциальных уравнений. Ред. Дж.Холл и Дж.Уатт. М.:Мир. 1979.-312 с.

[5] Ортега Дж., Пул У. Введение в численные методы решения дифференциальных уравнений. М.:Наука, 1986. -288 с.

[6] Дж.Форсайт, М.Малькольм, К.Моулер. Машинные методы математических вычислений. М.:Мир. 1980.-275 с.

4 Приложение

4.1 Подпрограмма RKF45

RKF45 - подпрограмма для решения задач Коши для систем первого порядка, основанная на формулах Рунге-Кутты. Подпрограмма требует 6 вычислений правых частей для продвижения на шаг интегрирования. Вычисления проводятся по формулам

$$y_{n+1} = y_n + \sum_{j=1}^{6} \gamma_j k_j,$$

$$k_i = h_n f\left(y_n + \sum_{j=1}^{i-1} \beta_{ij} k_j,\ t_n + \alpha_i h_n\right),\ i = \overline{1,6}.$$

Приведенные формулы дают 5-ый порядок точности.

Формулы

$$y^*_{n+1} = y_n + \sum_{j=1}^{6} \gamma^*_j k_j$$

позволяют найти решение с 4-ым порядком точности. Значения коэффициентов приведены в таблице 2.

Таблица 2.

	α	β					γ	γ^*
		1	2	3	4	5		
1	0						$\frac{16}{135}$	$\frac{25}{216}$
2	$\frac{1}{4}$	$\frac{1}{4}$					0	0
3	$\frac{3}{8}$	$\frac{3}{12}$	$\frac{9}{32}$				$\frac{6656}{12825}$	$\frac{1408}{2565}$
4	$\frac{12}{13}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$			$\frac{28561}{56430}$	$\frac{2197}{4104}$
5	1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$		$-\frac{9}{50}$	$-\frac{1}{5}$
6	$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	$\frac{2}{55}$	0

Подпрограмма в действительности не вычисляет значения y^*_{n+1}, а находит оценку $\sum_{i=1}^{6}(\gamma_i - \gamma_i^*)k_i$, используемую для контроля величины шага. В конце параграфа приводится иллюстрирующая программа с помощью которой рассчитывается движение двух тел под действием гравитационного притяжения.

Пусть $x(t)$ и $y(t)$ координаты одного тела в системе начало отсчета которой зафиксировано в другом теле. Уравнения движения в этом случае записываются в виде

$$x''(t) = -\frac{\alpha^2 x(t)}{R(t)},\ y''(t) = -\frac{\alpha^2 y(t)}{R(t)},$$
$$R(t) \equiv [x^2(t) + y^2(t)]^{3/2}.$$

Здесь α - константа, зависящая от гравитационной постоянной, масс обоих тел и выбранных единиц измерения. Если начальные условия взять в виде

$$x(0) = 1 - e,\ x'(0) = 0,$$
$$y(0) = 0,\ y'(0) = \alpha\sqrt{\frac{1+e}{1-e}},$$

где $e \in [0, 1]$ - параметр, то решение оказывается периодическим с периодом $2\pi/\alpha$. Орбита в этом случае будет эллипсом с эксцентриситетом e, один из фокусов которого находится в начале координат.

Для преобразования задачи к системе первого порядка, положим

$$y_1 = x,\ y_2 = y,\ y_3 = x',\ y_4 = y'.$$

Эквивалентная задача для системы первого порядка имеет вид

$$\begin{cases} y'_1(t) = y_3(t),\ y_1(0) = 1 - e \\ y'_2(t) = y_4(t),\ y_2(0) = 0 \\ y'_3(t) = -\frac{y_1(t)}{R(t)},\ y_3(0) = 0 \\ y'_4(t) = -\frac{y_2(t)}{R(t)},\ y_{4(}0) = \alpha\sqrt{\frac{1+e}{1-e}} \\ R(t) \equiv \frac{\sqrt{(y_1^2+y_2^2)^3}}{\alpha^2} \end{cases}$$

Описание подпрограммы приводится в ее тексте. Отметим лишь роль параметра IFLAG. При первом обращении к подпрограмме RKF45 параметру IFLAG присваивается значение равное единице. RKF45 изменяет его значение на 2 и при следующих обращениях необходимо сохранить это значение. Значения параметра IFLAG, не равные 2, сигнализируют о наличии различных нерегулярностей или ошибок. IFLAG=4 и IFLAG=7 - предупреждение, что прорамме трудно получить требуемую точность. Процесс можно продолжать, либо увеличить границы погрешностей. IFLAG=3 сигнализирует о слишком высокой затребованной относительной точности. IFLAG=5 или 6 означают, что для продолжения необходимо изменить допуски на ошибку. IFLAG=8 указывает на неправильность вызова RKF45. Отметим, что термины "число вычислений производных" или "число значений функции" в комментариях подпрограммы RKF45 следует понимать как число обращений к подпрограмме вычисления правых частей системы.

```
C
      SUBROUTINE ORBIT(T,Y,YP)
      REAL T,Y(4),YP(4),R,ALFASQ
      COMMON ALFASQ
      R=Y(1)*Y(1)+Y(2)*Y(2)
      R=R*SQRT(R)/ALFASQ
      YP(1)=Y(3)
      YP(2)=Y(4)
      YP(3)=-Y(1)/R
      YP(4)=-Y(2)/R
      RETURN
      END
C
      EXTERNAL ORBIT
      REAL T,Y(4),TOUT,RELERR,ABSERR
      REAL TFINAL,TPRINT,ECC,ALFA,ALFASQ,WORK(27)
      INTEGER IWORK(5),IFLAG,NEQN
      COMMON ALFASQ
      ECC=0.25
      ALFA=3.141592653589/4.0
      ALFASQ=ALFA*ALFA
      NEQN=4
      T=0.0
      Y(1)=1.0-ECC
      Y(2)=0.0
      Y(3)=0.0
      Y(4)=ALFA*SQRT((1.0+Ецц)/(1.0-Ецц))
      RELERR=1.0E-9
      ABSERR=0.0
      TFINAL=12.0
      TPRINT=0.5
      IFLAG=1
      TOUT=T
10    CALL rkf45(ORBIT,NEQN,Y,T,TOUT,RELERR,ABSERR,IFLAG,
     1          WORK,IWORK)
      WRITE(6,11) T,Y(1),Y(2)
      GO TO (80,20,30,40,50,60,70,80),IFLAG
20    TOUT=T+TPRINT
      IF(T.LT.TFINAL) GO TO 10
      STOP
30    WRITE(6,31) RELERR,ABSERR
      GO TO 10
40    WRITE(6,41)
      GO TO 10
50    ABSERR=1.0E-9
      WRITE(6,31) RELERR,ABSERR
      GO TO 10
60    RELERR=10.0*RELERR
      WRITE(6,31) RELERR,ABSERR
      IFLAG=2
      GO TO 10
```

```
70    WRITE(6,71)
      IFLAG=2
      GO TO 10
80    WRITE(6,81)
      STOP
11    FORMAT(F5.1,2F15.9)
31    FORMAT(17H TOLERANCES RESET,2E12.3)
41    FORMAT(11H MANY STEPS)
71    FORMAT(12H MUCH OUTPUT)
81    FORMAT(14H IMPROPER CALL)
      END
C
      SUBROUTINE RKF45(F,NEQN,Y,T,TOUT,RELERR,ABSERR,
     1          IFLAG,WORK,IWORK)
C
C     Метод Рунге-Кутты-Фельберга четвертого-пятого порядка
C     Составители программы-H.A.WATTS,L.F.SHAMPINE
C               SANDIA LABORATORIES
C               ALBUQUERQUE, NEW MEXICO
C
C     RKF45 предназначена главным образом для решения
C     нежестких и слабо жестких диф-х ур-ний, когда вычисление
C     производных не слишком дорогостоящее. RKF45, вообще говоря,
C     не следует использовать, если пользователю требуется
C     высокая точность.
C     Резюме:
C     подпрограмма RKF45 интегрирует систему из NEQN обыкновен-
C     ных диф. ур-ний первого порядка следующего вида:
C               DY(I)/DT=F(T,Y(1),Y(2),...,Y(NEQN)),
C     где Y(I) заданы в т. T.
C     обычно подпрограмму применяют для интегрирования от T до
C     TOUT, однако ее можно использовать и как одношаговый
C     интегратор, чтобы продолжить решение на один шаг в направлении TOUT
C     на выходе параметрам, фигурирующим в списке вызова, присваиваются
C     значения, необходимые для продолжения интегрирования. Пользовате-
C     лю нужно лишь еще раз обратиться к RKF45 (и, возможно, определить
C     новое значение для TOUT). В действительности RKF45 - это программа
C     интерфейса, которая вызывает подпрограмму RKFS, осуществляющую
C     процесс решения. RKFS в свою очередь вызывает подпрограмму
C     FEHL, которая вычисляет приближенное решение на один шаг.
C     RKF45 использует метод Рунге-Кутты-Фельберга, описанный в
C     следующей публикации: E.FEHLWERG,LOW-ORDER CLASSICAL RUNGE-
C     KUTTA FORMULAS WITH STEPSIZE CONTROL,NASA TR R-315.
C     Стиль работы пр-мы RKF45 иллюстрируется в следующих публика-
C     циях: L.F.SHAMPINE,H.A.WATTS,S.DAVENPORT,SOLVING NON-STIFF
C     ORDINARY DIFFERENTIAL EQUATIONS-THE STATE OF THE ART,SANDIA
C     LABORATORIES REPORT SAND75-0182,SIAM REVIEW,18(1976),N3,
C     376-411.
C
C     Параметры программы:
C     F - подпрограмма F(T/Y,YP) для вычисления производных;
C     YP(I)=DY(I)/DT;
C     NEQN - число интегрируемых ур-ний;
C     Y(*) - решение в точке T;
C     T - независимая переменная;
C     TOUT - точка выхода, в которой нужно определить значение реш-я;
C     RELERR,ABSERR - границы абсолютной и относительной погрешности
C     теста локальной ошибки. На каждом шаге пр-ма требует выпол-
C     нения условия   ABS(LOCAL ERROR).LE.RELERR*ABS(Y)+ABSERR
C     для каждой компоненты векторов локальной ошибки и решения;
C     IFLAG - указатель режима интегрирования;
C     WORK(*) - массив,содержащий информацию,внутреннюю для RKF45,
C     которая необходима при последующих вызовах. Его размерность
C     должна быть не меньше 3+6*NEQN;
C     IWORK(*) - целый массив,содержащий информацию, внутренюю для
C     RKF45, которая необходима при последующих вызовах. Его раз-
C     мерность должна быть не меньше 5.
C
C     Первое обращение к RKF45.
C
C     Пользователь должен предусмотреть в своей вызывающей программе
C     память для следующих массивов, фигурирующих в списке вызова
C     - Y(GNEQN),WORK(3+6*NEQN),IWORK(5); кроме того, он должен
C     об'явить F в операторе EXTERNAL, подготовить подпрограмму
C     F(T,Y,YP) и присвоить начальные значения параметрам -
C     NEQN - число интегрируемых ур-ний(NEQN.GE.1);
C     Y(*)-вектор начальных условий;
```

```
C     T-начальная точка интегрирования, T должно быть переменной;
C     TOUT - точка выхода, в которой нужно найти значение решения;
C     T=TOUT возможно лишь при первом обращении. В этом случае
C     выход из RKF45 происходит со значением параметра IFLAG=2,
C     если можно продолжать интегрирование;
C     RELERR,ABSERR - границы для относительной и абсолютной локаль-
C     ных погрешностей. Эти границы должны быть неотрицательны.
C     RELERR должна быть переменной, а ABSERR может быть и конс-
C     тантой. Программе,вообще говоря, не следует задавать грани-
C     цу для относительной ошибки,меньшую,чем примерно 1.e-8
C     дабы избежать трудностей,связанных с очень высокими запро-
C     сами к точности. Программа требует,чтобы RELERR была больше,
C     чем некоторый параметр относительной ошибки, вычисляемый
C     внутри ее и зависящий от машины. В частности, не разрешает-
C     ся задание только абсолютной ошибки. Если же задано значение
C     RELERR, меньшее допустимого, то RKF45 увеличивает RELERR на-
C     длежащим образом и возвращает управление пользователю, прежде
C     чем продолжать интегрирование.
C     IFLAG=+1,-1. Это - указатель настройки пр-мы для каждой новой
C     задачи. Нормальное входное значение равно +1. Пользователь
C     должен задавать IFLAG=-1 лишь в том случае, когда необходимо
C     управление одношаговым интегратором. В этом случае RKF45 пы-
C     тается продолжить решение на один шаг в напрвлении TOUT при
C     каждом очередном вызове. Поскольку этот режим работы весьма
C     неэкономичен, его следует применять лишь в случае крайней
C     необходимости.
C
C     Информация на выходе.
C
C     Y(*)-решение в точке T;
C     T - последняя точка, достигнутая при интегрировании;
C     IFLAG=2-при интегрировании достигнуто TOUT. Это значение
C     параметра указывает на успешный выход и является нормаль-
C     ным режимом для продолжения интегрирования;
C     IFLAG=-2 - был предпринят один шаг в напровлении TOUT, ока-
C     завшийся успешным. Это нормальный режим для продолжения
C     пошагового интегрирования;
C     IFLAG=3 - интегрирование не было закончено из-за того, что
C     заданное значение границы слишком мало. Для продолжения
C     интегрирования RELERR было надлежащим образом увеличено;
C     IFLAG=4 - интегрирование не было закончено из-за того, что
C     потребовалось более 3000 вычислений производной. Это соот-
C     ветствует приблизительно 500 шагам;
C     IFLAG=5 - интегрирование не было закончено из-за того, что
C     решение обратилось в нуль, вследствие чего тест только отно-
C     сительной ошибки не проходит. Для продолжения необходимо
C     ненулевое значение параметра ABSERR. Использование на
C     один шаг режима пошагового интегрирования является разумным
C     выходом из положения;
C     IFLAG=6 - интегрирование не было закончено из-за того, что
C     требуемая точность не могла быть достигнута даже при наи-
C     меньшей допустимой длине шага. Пользователь должен увели-
C     чить границу погрешности,прежде чем можно будет попытаться
C     продолжать интегрирование;
C     IFLAG=7 - по всей видимости, RKF45 неэффективна при решении
C     этой задачи. Слишком большое число требуемых выходных точек
C     препятствует выбору естественной величины шага. Следует ис-
C     пользовать режим пошагового интегрирования;
C     IFLAG=8 - неправильное задание входных параметров. Это зна-
C     чение появляется, если допущена одна из следующих ошибок:
C               NEQN.LE.0
C               T=TOUT и IFLAG.NE.+1 или-1
C               RELERR или ABSERR.LT.0
C               IFLAG.EQ.0 или .LT.-2 или .GT.8
C     WORK(*),IWORK(*) - информация,которая обычно не представляет
C     интереса для пользователя, но необходима при последующих
C     вызовах. WORK(1),...,WORK(NEQN) содержат первые производные
C     вектора решения Y в точке т.WORK(NEQN+1) хранит величину
C     шага H, с которой можно попытаться провести следующий шаг.
C     В IWORK(1) содержится счетчик числа вычислений производных.
C     Последующие обращения к RKF45.
C     На выходе подпрограммы RKF45 имеется вся информация, необ-
C     ходимая для продолжения интегрирования. Если при интегриро-
C     вании достигнуто TOUT, то пользователю достаточно определить
```

```
C     новое значение TOUT и снова обратиться к RKF45. В режиме по-
C     шагового интегрирования (IFLAG=-2) пользователь должен иметь
C     в виду, что каждый шаг выполняется в направлении текущего
C     значения TOUT. По достижении TOUT (сигнализируемом изменени-
C     ем IFLAG на 2) пользоыватель должен задать новое значение
C     TOUT и переопределить IFLAG на -2, чтобы продолжать в режиме
C     пошагового интегрирования.
C     Если интегрирование не было закончено, но пользователь хочет
C     продолжать (случай IFLAG=3,4), он попросту снова обращается
C     к RKF45. при IFLAG=3 параметр RELERR был изменен надлежащим
C     для продолжения интегрировавния образом. В случае IFLAG=4
C     счетчик числа значений функции будет переопределен на 0,
C     и будут разрашаны еще 3000 вычислений функции.
C     Однако в случае IFLAG=5,прежде чем можно будет продолжать
C     интегрирование, пользователь должен сначала изменить крите-
C     пий ошибки, задав положительное значение для ABSERR. Если
C     он не сделает это, выполнение пр-мы будет прекращено.
C     точно так же, в случае IFLPG=6, прежде чем продолжать инте-
C     грирование, пользователю необходимо переопределить IFLAG
C     на 2 (или -2, если использовать режим пошагового интегриро-
C     вания) и увеличить значение для ABSERR либо RELERR, либо и
C     для того, и для другого. Если это не будет сделано, выпол-
C     нение пр-мы прекращается. Появление IFLAG=6 указывает на
C     нерегулярность (решение быстро меняется или,возможно, имеет-
C     ся особенность), и часто в подобных случаях не имеет смысла
C     продолжать интегрирование.
C     Если будет получено значение IFLAG=7, то пользователь должен
C     перейти к режиму пошагового интегрирования с величиной шага,
C     определяемой пр-мой, или рассмотреть возможность перехода на
C     программы методов Адамса. Если все же пользователь хочет про-
C     должать интегрирование по подпрограмме RKF45, он должен до
C     нового обращения к ней переопределить IFLAG на 2. В против-
C     ном случае выполнение пр-мы будет прекращено.
C     Если получено значение IFLAG=8, то интегрирование нельзя про-
C     должать, пока не будут исправлены ошибочные входные параметры.
C        Нужно отметить, что массивы WORK и IWORK содержат информа-
C     цию, необходимую для дальнейшего интегрирования. Поэтому в
C     эти массивы нельзя вносить изменений.
      INTEGER NEQN,IFLAG,IWORK(5)
      REAL Y(NEQN),T,TOUT,RELERR,ABSERR,WORK(1)
C     Если транслятор проверяет индексы, то заменить WORK(1) на
C     WORK(3+6*NEQN)
      EXTERNAL F
      INTEGER K1,K2,K3,K4,K5,K6,K1M
C     Вычислить индексы для расщепления рабочего массива
      K1M=NEQN+1
      K1=K1M+1
      K2=K1+NEQN
      K3=K2+NEQN
      K4=K3+NEQN
      K5=K4+NEQN
      K6=K5+NEQN
C     Эта промежуточная программа просто сокращает для пользовате-
C     ля длинный список вызова путем пасщепления двух рабочих мас-
C     сивов. Если это не совместимо с транслятором, который имеется
C     в распоряжении пользователя, то он должен обращаться непо-
C     средственно к подпрограмме RKFS.
      CALL RKFS(F,NEQN,Y,T,TOUT,RELERR,ABSERR,IFLAG,
     1          WORK(1),WORK(K1M),WORK(K1),WORK(K2),
     2         WORK(K3),WORK(K4),WORK(K5),WORK(K6),
     3          WORK(K6+1),IWORK(1),IWORK(2),IWORK(3),
     4          IWORK(4),IWORK(5))
      RETURN
      END
C
      SUBROUTINE RKFS(F,NEQN,Y,T,TOUT,RELERR,ABSERR,IFLAG,
     1          YP,H,F1,F2,F3,F4,F5,SAVRE,SAVAE,NFE,KOP,INIT,
     2          JFLAG,KFLAG)
C
C     Метод Рунге-Кутта-Фельберга четвертого-пятого порядка
C     RKFS интегрирует систему обыкновенных диф. ур-ний первого
C     порядка (см. комментарий к RKF45). Массивы YR,F1,F2,F3,F4,
C     F5 (размерности по крайней мере NEQN) и переменные H,SAVRE,
C     SAVAE,NFE,KOP,INIT,JFLAG и KFLAG используются внутри программы
C     и вынесены в список вызова, чтобы сохранить их определен-
```

```
C     ность при повторном обращении. Поэтому их значения не дол-
C     жны изменяться пользователем. Возможный интерес представля-
C     ют параметры:
C     YP - производная вектора решения в точке T;
C     H - предполагаемый размер шага для очередного этапа;
C     NFE - счетчик числа вычислений функции;
      LOGICAL HFAILD,OUTPUT
      INTEGER NEQN,IFLAG,NFE,KOP,INIT,JFLAG,KFLAG
      REAL Y(NEQN),T,TOUT,RELERR,ABSERR,H,YP(NEQN),F1(NEQN),
     1          F2(NEQN),F3(NEQN),F4(NEQN),F5(NEQN),
     2          SAVRE,SAVAE
      EXTERNAL F
      REAL A,AE,DT,EE,EEOET,ESTTOL,ET,HMIN,REMIN,RER,S,SCALE,
     1          TOL,TOLN,U26,EPSP1,EPS,YPK
      INTEGER K,MAXNFE,MFLAG
      REAL AMAX1,AMIN1
C
C     REMIN - это минимальное допустимое значение для RELERR. По-
C     пытки получить по этой подпрограмме более высокую точность
C     обычно стоят очень дорого и зачастую безуспешны.
      DATA REMIN/1.E-12/
C
C     Стоимость счета контролируется требованием, чтобы количество
C     вычислений функции было ограничено величиной, приблизитель-
C     но равной значению параметра MAXNFE. Принятое здесь значе-
C     ние примерно соответствует 500 шагам.
      DATA MAXNFE/3000/
C
C     Проверить входные параметры
      IF(NEQN.LT.1) GO TO 10
      IF((RELERR.LT.0.0).OR.(ABSERR.LT.0.0)) GO TO 10
      MFLAG=IABS(IFLAG)
      IF((MFLAG.EQ.0).OR.(MFLAG.GT.8)) GO TO 10
      IF(MFLAG.NE.1) GO TO 20
C
C     Первый вызов, вычислить машинное эпсилон
      EPS=1.0
5     EPS=EPS/2.0
      EPSP1=EPS+1.0
      IF(EPSP1.GT.1.0) GO TO 5
      U26=26.0*EPS
      GO TO 50
C
C     Ошибка во входной информации
10    IFLAG=8
      RETURN
C
C     Проверить возможность продолжения
20    IF((T.EQ.TOUT).AND.(KFLAG.NE.3)) GO TO 10
      IF(MFLAG.NE.2) GO TO 25
C
      IF((KFLAG.EQ.3).OR.(INIT.EQ.0)) GO TO 45
      IF(KFLAG.EQ.4) GO TO 40
      IF((KFLAG.EQ.5).AND.(ABSERR.EQ.0.0)) GO TO 30
      IF((KFLAG.EQ.6).AND.(RELERR.LE.SAVRE).AND.
     1          (ABSERR.LE.SAVAE)) GO TO 30
      GO TO 50
C
25    IF(IFLAG.EQ.3) GO TO 45
      IF(IFLAG.EQ.4) GO TO 40
      IF((IFLAG.EQ.5).AND.(ABSERR.GT.0.0)) GO TO 45
C
C     Интегрирование нельзя продолжать, поскольку пользователь
C     не выполнил инструкций, соответствующих значениям
C     IFLAG=5,6,7 или 8
30    STOP
C
C     Переопределить счетчик числа вычислений функции
40    NFE=0
      IF(MFLAG.EQ.2) GO TO 50
C
C     Переопределить значение FLAG, установленное при предыдущем
C     обращении
45    IFLAG=JFLAG
      IF(KFLAG.EQ.3) MFLAG=IABS(IFLAG)
C
C     Сохранить входное значение IFLAG и установить значение FLAG,
C     соответствующее продолжению, для будующей входной проверки
50    JFLAG=IFLAG
      KFLAG=0
C
```

```
C     Сохранить значения RELERR и ABSERR для входной проверки
C     при последующих обращениях
      SAVRE=RELERR
      SAVAE=ABSERR
C
C     Установить значение границы для относительной погрешности,
C     равное как минимум 2*EPS+REMIN, чтобы избежать трудностей,
C     связанных с тредованием недостижимой точности
      RER=2.0*EPS+REMIN
      IF(RELERR.GE.RER) GO TO 55
C
C     Заданная граница относительной погрешности слишком мала
      RELERR=RER
      IFLAG=3
      KFLAG=3
      RETURN
C
55    DT=TOUT-T
      IF(MFLAG.EQ.1) GO TO 60
      IF(INIT.EQ.0) GO TO 65
      GO TO 80
C
C     Присвоение начальных значений (инициирование) - установить
C     значение указателя окончания интегрирования, INIT;
C     установить значение указателя слишком большого затребован-
C     ного числа выходных точек, KOP;
C     вычислить начальные производные;
C     установить значение счетчика числа вычислений функции,NFE;
C     оценить начальную величину шага;
60    INIT=0
      KOP=0
      A=T
      CALL F(A,Y,YP)
      NFE=1
      IF(T.NE.TOUT) GO TO65
      IFLAG=2
      RETURN
C
65    INIT=1
      H=ABS(DT)
      TOLN=0.
      DO 70 K=1,NEQN
                TOL=RELERR*ABS(Y(K))+ABSERR
                IF(TOL.LE.0.0) GO TO 70
                TOLN=TOL
                YPK=ABS(YP(K))
                IF(YPK*H**5.GT.TOL) H=(TOL/YPK)**0.2
70    CONTINUE
      IF(TOLN.LE.0.0) H=0.0
      H=AMAX1(H,U26*AMAX1(ABS(T),ABS(DT)))
      JFLAG=ISIGN(2,IFLAG)
C
C     Присвоить величине шага знак, соответствующий интегрирова-
C     нию в направлении от T к TOUT
80    H=SIGN(H,DT)
C
C     Проверка, насколько серьезно влияние на RKF45 слишком боль-
C     шого затребованного числа входных точек
      IF(ABS(H).GE.2.0*ABS(DT)) KOP=KOP+1
      IF(KOP.NE.100) GO TO 85
C
C     Чрезмерная частота выходов
      KOP=0
      IFLAG=7
      RETURN
C
85    IF(ABS(DT).GT.U26*ABS(T)) GO TO 95
C
C     Если очень близко к точке выхода, экстраполировать и
C     вернуться по месту вызова
      DO 90 K=1,NEQN
90    Y(K)=Y(K)+DT*YP(K)
      A=TOUT
      CALL F(A,Y,YP)
      NFE=NFE+1
      GO TO 300
C
C     Присвоить начальное значение индикатору точки выхода
95    OUTPUT=.FALSE.
C
C     Чтобы избежать неоправданного машинного нуля при вычислении
C     функции от границ погрешности, промасштабировать эти
C     границы
      SCALE=2.0/RELERR
```

```
      AE=SCALE*ABSERR
C
C     Пошаговое интегрирование
100   HFAILD=.FALSE.
C
C     установить наименьшую допустимую величину шага
      HMIN=U26*ABS(T)
C
C     Исправить при необходимости величину шага, чтобы достигнуть
C     точки выхода. Рассчитать на два шага вперед, чтобы избежать
C     слишком резких изменений в величине шага и тем самым умень-
C     шить влияние выходных точек на программу.
      DT=TOUT-T
      IF(ABS(DT).GE.2.0*ABS(H)) GO TO 200
      IF(ABS(DT).GT.ABS(H)) GO TO 150
C
C     Следующий успешный шаг завершит интегрирование до указанной
C     точки выхода
      OUTPUT=.TRUE.
      H=DT
      GO TO 200
150   H=0.5*DT
C
C     Внутренний одношаговый интегратор.
C
C     Границы погрешностей были промасштабированы, чтобы избежать
C     неоправданного машинного нуля при вычислении функции ET от
C     них. Чтобы избежать обращения в нуль знаменателя в тексте,
C     относительная ошибка измеряется по отношению к среднему из
C     величин решения в начале и конце шага. В формуле, оценивающ-
C     щей ошибку, произведена группировка слагаемых, уменьшающая
C     потерю верных знаков. Чтобы различать между собой разные
C     аргументы, для H не допускаются значения, меньшие умножен-
C     ной на 26 ошибки округления в T. Введены практические огра-
C     ничения на скорость изменения величины шага, чтобы сгладить
C     процесс выбора этой величины и избежать чрезмерного ее раз-
C     броса в задачах с нарушением непрерывности.
C     Из предосторожности программа берет 9/10 от той величины
C     шага, которая нужна по ее оценке. Если на данном шаге была
C     неудачная попытка, то при планировании следующего увеличение
C     длины шага не допускается. Это повышает эффективность прог-
C     раммы для задач с разрывами и в общем случае, поскольку ис-
C     пользуется локальная экстраполяция и дополнительная предос-
C     торожность кажется оправданной.
C
C     Проверить число вычислений производных. Если оно не превы-
C     шает установленного предела, попробовать продолжить интегри-
C     рование с T до T+H
200   IF(NFE.LE.MAXNFE) GO TO 220
C
C     Слишком большая работа
      IFLAG=4
      KFLAG=4
      RETURN
C
C     Продолжить приближенное решение на один шаг длины H
220   CALL FEHL(F,NEQN,Y,T,H,YP,F1,F2,F3,F4,F5,F1)
      NFE=NFE+5
C
C     Вычислить и сравнить допустимые границы и оценки локальной
C     ошибки, а затем снять масштабирование границ. Заметьте, что
C     относительная ошибка измеряется по отношению к среднему из
C     величин решения в начале и конце шага.
      EEOET=0.0
      DO 250 K=1,NEQN
                ET=ABS(Y(K))+ABS(F1(K))+AE
                IF(ET.GT.0.0) GO TO 240
C
C     Неправильная граница погрешности
                IFLAG=5
                RETURN
240   EE=ABS((-2090.0*YP(K)+(21970.0*F3(K)-15048.0*F4(K)))
     1          +(22528.0*F2(K)-27360.0*F5(K)))
250   EEOET=AMAX1(EEOET,EE/ET)
      ESTTOL=ABS(H)*EEOET*SCALE/752400.0
      IF(ESTTOL.LE.1.0) GO TO 260
C
C     Неудачный шаг. Уменьшить величину шага и снова попробовать.
C     уменьшение ограничивается снизу множителем 1/10.
      HFAILD=.TRUE.
      OUTPUT=.FALSE.
```

```
      S=0.1
      IF(ESTTOL.LT.59049.0) S=0.9/ESTTOL**0.2
      H=S*H
      IF(ABS(H).GT.HMIN) GO TO 200
C
C     Заданная граница ошибки недостижима даже при наименьшей
C     допустимой величине шага
      IFLAG=6
      KFLAG=6
      RETURN
C
C     Успешный шаг. Поместить в массив Y решение в точке T+H и
C     вычислить производные в этой точке.
260   T=T+H
      DO 270 K=1,NEQN
270   Y(K)=F1(K)
      A=T
      CALL F(A,Y,YP)
      NFE=NFE+1
C
C     Выбрать величину следующего шага. Увеличение ограничено
C     множителем 5. Если на данном шаге была неудачная попытка,
C     то для следующего не допускается выбор большей величины
C     шага.
      S=5.0
      IF(ESTTOL.GT.1.889568E-4) S=0.9/ESTTOL**0.2
      IF(HFAILD) S=AMIN1(S,1.0)
      H=SIGN(AMAX1(S*ABS(H),HMIN),H)
C
C     Конец одношагового интегратора.
C
C     Нужно ли делать очередной шаг
      IF(OUTPUT) GO TO 300
      IF(IFLAG.GT.0) GO TO 100
C
C     Интегрирование успешно завершено
C
C     Режим одношагового интегрирования
      IFLAG=-2
      RETURN
C
C     Режим интегрирования на интервале
300   T=TOUT
      IFLAG=2
      RETURN
      END
C
      SUBROUTINE FEHL(F,NEQN,Y,T,H,YP,F1,F2,F3,F4,F5,S)
C
C     Метод Рунге-Кутта-Фельберга четвертого-пятого порядка.
C
C     Подпрограмма FEHL интегрирует систему из NEQN обыкновенных
C     диф. ур-ний первого порядка следующего вида
C               DY(I)/DT=F(T,Y(1),...,Y(NEQN)),
C     где начальные значения Y(I) и начальные производные YP(I)
C     заданы в начальной точке T. FEHL продолжает решение на
C     фиксированный шаг H и помещает в массив S(I) приближение
C     к решению в точке T+H, имеющее пятый порядок точности
C     (локальный порядок равен 6). F1,...,F5- массивы размер-
C     ности NEQN, необходимые внутри программы.
C     В формулах произведена группировка с целью уменьшить поте-
C     рю верных знаков.
C     Чтобы можно было различать разные независимые аргументы,
C     при обращении к FEHL не следует задавать для N значение,
C     меньшее умноженной на 13 ошибки округления в T.
      INTEGER NEQN
      REAL Y(NEQN),T,H,YP(NEQN),F1(NEQN),F2(NEQN),F3(NEQN),
     1          F4(NEQN),F5(NEQN),S(NEQN)
      REAL CH
      INTEGER K
      CH=H/4.0
      DO 221 K=1,NEQN
221   F5(K)=Y(K)+CH*YP(K)
      CALL F(T+CH,F5,F1)
      CH=3.0*H/32.0
      DO 222 K=1,NEQN
222   F5(K)=Y(K)+CH*(YP(K)+3.0*F1(K))
      CALL F(T+3.0*H/8.0,F5,F2)
      CH=H/2197.0
      DO 223 K=1,NEQN
223   F5(K)=Y(K)+CH*(1932.0*YP(K)+(7296.0*F2(K)-
```

```
     1          7200.0*F1(K)))
      CALL F(T+12.0*H/13.0,F5,F3)
      CH=H/4104.0
      DO 224 K=1,NEQN
224   F5(K)=Y(K)+CH*((8341.0*YP(K)-845.0*F3(K))+
     1          (29440.0*F2(K)-32832.0*F1(K)))
      CALL F(T+H,F5,F4)
      CH=H/20520.0
      DO 225 K=1,NEQN
225   F1(K)=Y(K)+CH*((-6080.0*YP(K)+(9295.0*F3(K)-
     1          5643.0*F4(K)))+(41040.0*F1(K)-28352.0*F2(K)))
      CALL F(T+H/2.0,F1,F5)
C
C     Вычислить приближенное решение в точке T+H
      CH=H/7618050.0
      DO 230 K=1,NEQN
230   S(K)=Y(K)+CH*((902880.0*YP(K)+(3855735.0*F3(K)-
     1          1371249.0*F4(K)))+(3953664.0*F2(K)+
     2          277020.0*F5(K)))
      RETURN
      END
```

Аналогичная иллюстрирующая программа с помощью которой рассчитывается движение двух тел под действием гравитационного притяжения в системе Matlab приведена ниже. В результате ее выполнения получим на фозовой плоскости эллипс с эксцентриситетом e, один из фокусов которого находится в начале координат (см. рис.1).

```
function orbit
global alpha;
alpha=1;
e=0.99;
y0=[1-e; 0; 0; alpha*sqrt( (1+e)/(1-e))];

[t,z]=ode45(@f,[0 2.1*pi],y0);
plot(z(:,1),z(:,2))
xlabel ('x')
ylabel('y')
title('e=0.99,  \alpha=1')
grid
 function  z=f(t,y)
global alpha;
R=sqrt((y(1)^2+y(2)^2)^3)/alpha^2;
z(1)=y(3);
z(2)=y(4);
z(3)=-y(1)/R;
z(4)=-y(2)/R;
z=z';
return
```

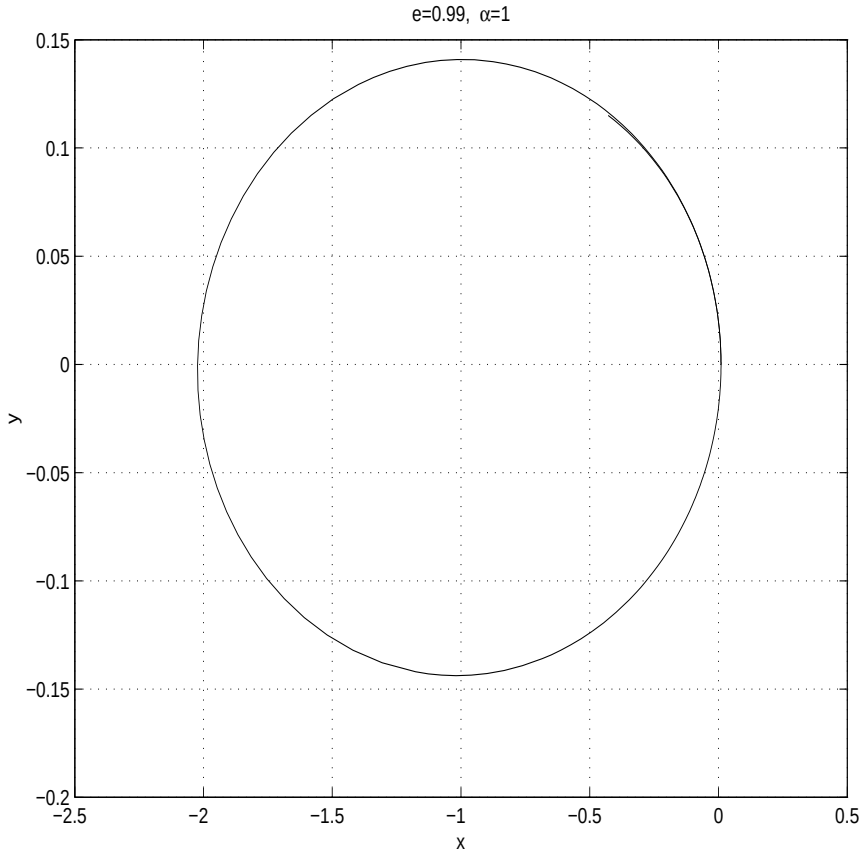

Рис. 1: Траектория движения на фазовой плоскости движения двух тел под действием гравитационного притяжения

ЗАДАЧА КОШИ ДЛЯ ОБЫКНОВЕННЫХ
ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ.
ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ

Составители:
Кулинич
Виктор Валентинович
Смирнов
Иван Паисьевич

Подписано к печати . Формат 60x84 1/16.
Печать офсетная. Бумага оберточная. Усл.печ.л. 1.7.

Тираж 500 экз. Заказ . Бесплатно.
Нижегородский государственный университет им.Н.И.Лобачевского.
603600 ГСП-20, Н.Новгород, просп.Гагарина, 23.
Типография ННГУ. 603600, Н.Новгород, ул.Б.Покровская, 37.