УДК 519.62:517.93:521.1

РЕАЛИЗАЦИЯ МЕТОДА РЯДОВ ТЕЙЛОРА ДЛЯ РЕШЕНИЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Л. К. Бабаджанянц[1], **А. И. Большаков**[1]

Предлагаются новые алгоритм и программа явного метода рядов Тейлора с переменной величиной шага и порядка, ориентированные на решение нежестких обыкновенных дифференциальных уравнений (ОДУ) с полиномиальными правыми частями. Используемая версия метода основана на новых простых рекуррентных формулах вычисления коэффициентов рядов Тейлора и новых строгих априорных оценках локальной ошибки в сочетании с обычными нестрогими апостериорными соображениями. Программа авторов, написанная на Фортране 95, сравнивается с тремя известными программами, которые реализуют соответственно явные методы Дормана–Принса, Грегга–Булирша–Штера и рядов Тейлора. Численные эксперименты показали конкурентоспособность предлагаемой программы, ее применимость и надежность в реальных задачах динамики.

Ключевые слова: полиномиальная система ОДУ, метод рядов Тейлора, динамика.

Введение. В рамках реализации метода рядов Тейлора (МРТ) для нежестких полиномиальных систем обыкновенных дифференциальных уравнений (ОДУ) [1] предлагается алгоритм этого метода и соответствующая программа на Фортране 95. Заметим, что методом дополнительных переменных (МДП) многие более общие системы обыкновенных дифференциальных уравнений сводятся к рассматриваемым здесь полиномиальным системам: для применимости МДП необходимо и достаточно, чтобы правые части исходных уравнений можно было записать при помощи конечного числа операций алгебры и конечных суперпозиций функций, которые сами являются решениями полных систем уравнений в частных производных с полиномиальными правыми частями [2, 3] (система ОДУ является частным случаем полной системы).

Отметим, что существуют и развиваются и другие варианты МРТ, в которых используются иные идеи и средства, в частности: автоматическое дифференцирование [4–10]; итерации Пикара [11–14]; приближенное вычисление производных [15, 16]; интервальный анализ [17, 18]. Существуют также и программные реализации МРТ различных авторов, например ATOMFT [5, 19], COSY INFINITY [20, 21], TAYLOR [22], DAETS [23–25], TIDES [9, 26].

В первом разделе настоящей статьи приводятся используемые здесь сведения о МРТ из статьи [1], а во втором излагается алгоритм МРТ с автоматическим выбором величины шага и порядка. В третьем разделе рассматриваются следующие примеры: простейшее квадратичное уравнение, уравнения Эйлера для эллиптических функций Якоби, уравнения Лоренца, уравнения орбитального движения пяти внешних планет Солнечной системы при реальных значениях масс и начальных данных и уравнения задачи двух тел для каждой из этих планет при тех же данных. Представленные в форме таблиц и графиков результаты численных экспериментов получены при помощи нашей программы TSMR и трех других программ численного интегрирования DOP853 [27], ODEX [27, 28] и TIDES [9, 26], что позволит читателю составить свое представление о недостатках и достоинствах предлагаемых метода, алгоритма и программы. В заключении кратко обсуждаются результаты статьи. Программа с инструкцией и примерами ее применения доступна по адресу [29].

1. Метод рядов Тейлора. Здесь рассматриваются две формы задания полиномиальной задачи Коши, формулы для коэффициентов Тейлора и различные оценки локальных решений [1].

1.1. Задача Коши. Рассмотрим автономную задачу Коши

$$\frac{dx}{dt} = f(x), \quad x(t_0) = x_0, \tag{1}$$

[1] Санкт-Петербургский государственный университет, факультет прикладной математики — процессов управления, Университетский пр., 35, 198504, Петергоф, г. Санкт-Петербург; Л. К. Бабаджанянц, профессор, e-mail: levon@mail.wplus.net; А. И. Большаков, аспирант, e-mail: alexey.bolshakov@gmail.com

где $f = (f_1, \ldots, f_n)$, $x = (x_1, \ldots, x_n)$, $x_0 = (x_{10}, \ldots, x_{n0}) \in R^n$, t, $t_0 \in R$. Решение ее обозначим $x(t, t_0, x_0)$, $x(t)$, или x. Полиномиальной называют систему (1) (и задачу Коши), в которой все функции f_i — алгебраические полиномы по $x_1, \ldots, x_n$. Далее используются две формы задания полиномиальной системы.

Первая форма представления полиномиальной задачи Коши имеет вид

$$\frac{dx}{dt} = a + \sum_{m=1}^{L+1} \sum_{i \in I(m)} a[i]x^i, \quad x(t_0) = x_0, \tag{2}$$

где $i = (i_1, \ldots, i_n)$, $i_1, \ldots, i_n \in Z$, $L \in [0 : \infty)$, $I(m) = \Big\{ i \in Z^n \big| i_1, \ldots, i_n \geqslant 0, |i| = m \Big\}$, $|i| = i_1 + \ldots + i_n$, $x = (x_1, \ldots, x_n) \in R^n$, $x^i = x_1^{i_1} \ldots x_n^{i_n}$, $x_0 = (x_{10}, \ldots, x_{n0}) = \big(x_1(t_0), \ldots, x_n(t_0)\big) \in R^n$, $a = (a_1, \ldots, a_n) \in R^n$, $a[i] = \big(a_1[i], \ldots, a_n[i]\big) \in R^n$, t, $t_0 \in R$.

Вторая форма представления задачи (1) следующая:

$$\frac{dx_j}{dt} = \sum_{k=0}^{u} a_{jk} x^{i(k)}, \quad x_j(t_0) = x_{j0}, \quad j \in [1 : n], \tag{3}$$

где $x^{i(0)} = 1$, $x^{i(1)} = x_1$, …, $x^{i(n)} = x_n$, а $x^{i(n+1)}, \ldots, x^{i(u)}$ — все различные нелинейные мономы в правых частях уравнений (2).

1.2. Схемы и коэффициенты Тейлора. Набор $T = \big(x^{i(1)}, \ldots, x^{i(n)}, x^{i(n+1)}, \ldots, x^{i(N)}\big)$ считаем упорядоченным так, что $2 \leqslant \big|i(n+1)\big| \leqslant \big|i(n+2)\big| \leqslant \ldots \leqslant \big|i(N)\big| \leqslant L+1$.

Если любой моном $x^{i(r)}$, $\big|i(r)\big| \geqslant 3$ в T равен $x^{i(p)}\, x^{i(q)}$ при $1 \leqslant p < r$, $1 \leqslant q < r$, то вводят в рассмотрение *схему* $S = \Big(\big(p(n+1), q(n+1)\big), \ldots, \big(p(N), q(N)\big)\Big)$ из $N - n$ пар натуральных чисел $p(r)$ и $q(r)$, таких, что $r > p(r)$, $q(r)$ для любого $r \in [n+1 : N]$. При помощи схемы можно решить задачу последовательного вычисления всех мономов $x^{i(n+1)}, \ldots, x^{i(N)}$ набора T в предположении, что известны первые n его мономов $x^{i(1)} = x_1, \ldots, x^{i(n)} = x_n$.

Любой набор мономов можно дополнить новыми мономами так, чтобы он имел схему. Используя схему $S = \Big(\big(p(n+1), q(n+1)\big), \ldots, \big(p(u), q(u)\big)\Big)$ для набора всех различных нелинейных мономов в правых частях уравнений (3) (дополненного, если необходимо), для быстрого вычисления коэффициентов Тейлора x_{kp} их решения $x^{i(k)} = \sum\limits_{p=0}^{\infty} x_{kp}(t - t_0)^p$, $k \in [0 : u]$ можно вывести следующие рекуррентные формулы:

$$x_{k0} = x_k(t_0), \quad k \in [1 : n], \tag{4}$$

$$\left.\begin{aligned} x_{kp} &= \sum_{l=0}^{p} x_{p(k),l}\, x_{q(k),p-l}, && k \in [n+1 : u], \\ x_{k,p+1} &= (p+1)^{-1} \sum_{l=0}^{u} a_{kl} x_{lp}, && k \in [1 : n], \end{aligned}\right\} \quad p = 0, 1, \ldots . \tag{5}$$

1.3. Формулировка МРТ и оценки. Здесь рассматривается формулировка метода рядов Тейлора и используемые в его реализации оценки радиуса сходимости и остаточного члена для линейной и нелинейной задачи (2).

1.3.1. Формулировка МРТ. Введем обозначения: $x^{(k)} = \dfrac{\partial^k x}{\partial t^k}$, $x_0^{(k)} = x^{(k)}(t_0)$, $|x| = \max\limits_{i \in [1:n]} |x_i|$, $O_\rho(t_0) = \Big\{ t \in C \big| |t - t_0| < \rho \Big\}$, $T_M x(t, t_0, x_0) = \sum\limits_{m=0}^{M} x_0^{(m)} \dfrac{(t - t_0)^m}{m!}$, $\delta T_M x(t, t_0, x_0) = x(t, t_0, x_0) - T_M x(t, t_0, x_0)$, где T_M и δT_M — операторы, которые решению $x(t, t_0, x_0)$ задачи (2) сопоставляют полином Тейлора $T_M x(t, t_0, x_0)$ и остаточный член $\delta T_M x(t, t_0, x_0)$ соответственно. Радиус сходимости ряда $T_\infty x(t, t_0, x_0)$ обозначим через $R(t_0, x_0)$.

Метод рядов Тейлора решения задачи Коши (2) заключается в построении таблицы приближенных значений $\widetilde{x}_w = \widetilde{x}(t_w)$ по формулам $\widetilde{x}_1 = T_{M_1} x(t_1, t_0, x_0), \ldots, \widetilde{x}_w = T_{M_w} x(t_w, t_{w-1}, \widetilde{x}_{w-1}), \ldots$, где $M_1, M_2, \ldots$ — натуральные числа, $t_1 = t_0 + h_1$, $t_2 = t_1 + h_2, \ldots$, $h_1, h_2, \ldots$ удовлетворяют неравенствам $|h_w| < R\big(t_{w-1}, \widetilde{x}_{w-1}\big)$. Вычисление каждого значения $\widetilde{x}(t_w)$ называют шагом метода, а число h_w — величиной шага.

1.3.2. Оценки. Чтобы автоматизировать выбор порядка M_k и величины шага h_k, можно воспользоваться оценками величин $R(t_0, x_0)$ и $\delta T_M x(t, t_0, x_0)$. В разделе 3.4 статьи [1] перечислены все предложенные в ней средства такой автоматизации; здесь приводятся (в удобной для алгоритмизации форме) те из них, которые будут использованы ниже, в разделе 2.

1.3.2.1. Оценки для линейной задачи. Рассмотрим задачу Коши

$$\begin{aligned} &\frac{dx}{dt} = a + Ax, \quad x(t_0) = x_0, \quad x = (x_1, \ldots, x_n) \in R^n, \\ &x_0 = (x_{1,0}, \ldots, x_{n,0}) \in R^n, \quad a = (a_1, \ldots, a_n) \in R^n, \quad A = (a_{i,j}), t, t_0, a_{i,j} \in R. \end{aligned} \tag{6}$$

Введем замену $x_j = \alpha_j y_j,\ j \in\ [1:n]$, зависящую от "масштабирующих множителей" $\alpha = (\alpha_1, \ldots, \alpha_n)$, $\alpha_1, \ldots, \alpha_n > 0$, и обозначения

$$\rho(\alpha) = \frac{1}{s(\alpha)}, \quad s(\alpha) = \max_{i\in\ [1:n]} s_i(\alpha), \quad s_i(\alpha) = \alpha_i^{-1} \sum_{j=1}^{n} \alpha_j |a_{ij}|, \quad |b| = \max_{i\in\ [1:n]} \alpha_i^{-1} |a_i|,$$

$$|y_0| = \max_{i\in\ [1:n]} \bigl(\alpha_i^{-1} |x_{i0}|\bigr), \quad T_M e^\tau = \sum_{m=0}^{M} \frac{\tau^m}{m!}, \quad u(\tau) = \delta T_M e^\tau = e^\tau - T_M e^\tau. \tag{7}$$

Утверждение 1. *Решение $x(t, t_0, x_0)$ задачи (6) удовлетворяет неравенству*

$$\bigl|\delta T_M x_i(t, t_0, x_0)\bigr| \leqslant \alpha_i \bigl(|y_0| + |b|\rho\bigr)\, \delta T_M e^{|t-t_0|/\rho}, \quad i \in\ [1:n], \quad t \in C.$$

Утверждение 2. *Пусть u^{-1} — функция, обратная и при $\tau > 0$, $\chi_1, \ldots, \chi_n$, $\varepsilon_1, \ldots, \varepsilon_n$ — положительные числа и $\varepsilon = \bigl(|y_0| + \rho|b|\bigr)^{-1} \min\limits_{i\in\ [1:n]} \left(\dfrac{\varepsilon_i \chi_i}{\alpha_i}\right)$. Тогда*

$$\bigl(|t - t_0| \leqslant \rho\, u^{-1}(\varepsilon)\bigr) \Rightarrow \Bigl(\bigl|\delta T_M x_i(t, t_0, x_0)\bigr| \leqslant \varepsilon_i \chi_i,\ i \in [1:n]\Bigr).$$

1.3.2.2. Оценки для нелинейной задачи. В задаче (2) положим, что $x_j = \alpha_j y_j,\ j \in\ [1:n]$, $\alpha = (\alpha_1, \ldots, \alpha_n),\ \alpha_1, \ldots, \alpha_n > 0$, и предположим дополнительно, что x_{j0} и α удовлетворяют неравенствам $0 < |x_{j0}| \leqslant \alpha_j,\ j \in\ [1:n]$. Введем обозначения:

$$\rho(\alpha) = \frac{1}{Ls(\alpha)}, \quad s(\alpha) = \max_{j\in\ [1:n]} s_j(\alpha), \quad s_j(\alpha) = \alpha_j^{-1} \left(|a_j| + \sum_{m=1}^{L+1} \sum_{|i|=m} \alpha^i \bigl|a_j\ [i]\bigr|\right),$$

$$O_\rho(t_0) = \Bigl\{t \in C \bigm| |t - t_0| < \rho\Bigr\}, \quad b(\tau) = (1-\tau)^{-1/L}, \quad T_M b(\tau) = \sum_{m=0}^{M} \prod_{l=0}^{m-1} \frac{(1/L + l)\tau^m}{m!}, \tag{8}$$

$$v(\tau) = \delta T_M b(\tau) = b(\tau) - T_M b(\tau), \quad \tau \in\ [0, 1).$$

Утверждение 3. *Решение $x(t, t_0, x_0)$ задачи (2) регулярно в круге $O_\rho(t_0)$ и удовлетворяет там неравенству $\bigl|\delta T_M x_j(t, t_0, x_0)\bigr| \leqslant \alpha_j \delta T_M b\left(\dfrac{|t - t_0|}{\rho}\right)$.*

Утверждение 4. *Если функция v^{-1} обратна v, а $\varepsilon = \min\limits_{i\in\ [1:n]} (\varepsilon_i \chi_i / \alpha_i)$, $\chi_i, \varepsilon_i > 0$, то*

$$\bigl(|t - t_0| \leqslant \rho\, v^{-1}(\varepsilon)\bigr) \Rightarrow \Bigl(\bigl|\delta T_M x_i(t, t_0, x_0)\bigr| \leqslant \varepsilon_i \chi_i,\ i \in\ [1:n]\Bigr).$$

Отметим, что далее мы используем эти утверждения при $\varepsilon_1 = \varepsilon_2 = \ldots = \varepsilon_n, \chi_1 = \alpha_1, \ldots, \chi_n = \alpha_n$.

2. Алгоритмы метода рядов Тейлора. В основе предлагаемой реализации МРТ для полиномиальных ОДУ лежат формулы (4) и (5) для коэффициентов Тейлора и новые алгоритмы автоматического выбора величины шага и порядка, опирающиеся как на утверждения 1–4, так и на обычно применяемые в численном анализе эвристические соображения [28, 30]. Мы изложим вначале некоторые вспомогательные алгоритмы, потом опирающиеся на них алгоритмы автоматического выбора величины шага и порядка, а затем алгоритм интегрирования полиномиальной системы ОДУ на заданном промежутке.

Далее величины $t = t_k,\ x = x^k = \widetilde{x}(t_k),\ h = h_k = t_{k+1} - t_k$ будут обозначать текущий узел, приближенное значение решения в нем и величину текущего шага, а ε, Δ — заданные пользователем допустимые

относительную и абсолютную погрешности решения на шаге. Кроме того, при фиксированных M, K, t_k, x^k величины $T_M x\big(t_{k+1}, t_k, x^k\big)$ и

$$\delta T_{MK} x\big(t_{k+1}, t_k, x^k\big) = T_{M+K} x\big(t_{k+1}, t_k, x^k\big) - T_M x\big(t_{k+1}, t_k, x^k\big),$$
$$\varepsilon_{MK}\big(t_{k+1}, t_k, x^k\big) = \Big|\delta T_{MK} x\big(t_{k+1}, t_k, x^k\big)\Big| \left(\Big|\big(T_M x\big(t_{k+1}, t_k, x^k\big)\Big| + \Delta\right)^{-1}$$

будем записывать упрощенно как $T(h)$, $\delta T(h)$, $\varepsilon(h)$.

2.1. Вспомогательные алгоритмы и данные.

2.1.1. Таблицы для вычисления значений функций u^{-1} и v^{-1} (утверждения 2 и 4). Таблицы содержатся в [29] и используются в алгоритме 2.1.2.

Для каждой пары $L = 0, \ldots, 99$, $M = 1, \ldots, 99$ таблицы содержат набор значений $\big(u(\tau_j),\, v(\tau_j)\big)$, упорядоченный по возрастанию $\tau_j = 0.01, 0.02, \ldots, 0.99$. Они вычислены при помощи программы `Wolfram Mathematica` [31] по формулам $u(\tau_j) = e^{\tau_j} - T_M e^{\tau_j}$, $v(\tau_j) = b(\tau_j) - T_M b(\tau_j)$ (см. (7), (8)). Из равенств $u^{-1}\big(u(\tau_j)\big) = \tau_j$ и $v^{-1}\big(v(\tau_j)\big) = \tau_j$ следует, что значения функций u^{-1}, v^{-1} в промежуточных точках можно получить интерполированием по значениям в ближайших узловых точках $\big(u(\tau_j),\, v(\tau_j)\big)$.

2.1.2. Априорный выбор шага. Величина x_0 (значение решения в узле, для которого выбирается шаг h), ε, Δ (допустимые относительная и абсолютная погрешности) считаются заданными. Вначале вычисляются масштабирующие множители α_i, $i \in [1:n]$ по процедуре `tsmr_alp` [29], если она задана пользователем, иначе при $i \in [1:n]$ $\alpha_i = \begin{cases} \max\limits_{j\in[1:n]} (|x_j(t_0)|), & \max\limits_{j\in[1:n]} \big|x_j(t_0)\big| \neq 0, \\ 1, & \max\limits_{j\in[1:n]} \big|x_j(t_0)\big| = 0. \end{cases}$

Затем вычисляется величина $\tau = \begin{cases} v^{-1}\left(\varepsilon \min\limits_{j\in[1:n]} \dfrac{\big|x_j(t_0)\big| + \Delta}{\alpha_j}\right) & \text{для нелинейной системы,} \\ u^{-1}\left(\varepsilon \min\limits_{j\in[1:n]} \dfrac{\big|x_j(t_0)\big| + \Delta}{\alpha_j}\right) & \text{для линейной системы,} \end{cases}$

u^{-1}, v^{-1} вычисляются интерполированием (см. п. 2.1.1), $\rho = \rho(\alpha)$ (см. пп. 1.3.2.1, 1.3.2.2) и $h = \tau\rho$.

2.1.3. Итеративный алгоритм коррекции шага. Введем следующие обозначения: D — натуральное число, например, $D = 5$, $d = h_0/D$, $s = \operatorname{sgn}\big(\varepsilon - \varepsilon(h_0)\big)$, $\sigma(i,h) = \begin{cases} i, & \operatorname{sgn}\big(\varepsilon - \varepsilon(h)\big) \leqslant 0, \\ i-1, & \operatorname{sgn}\big(\varepsilon - \varepsilon(h)\big) > 0, \end{cases}$ где h_0 — шаг, который требуется скорректировать так, чтобы получить максимально возможную его величину при заданной пользователем допустимой относительной погрешности ε.

Алгоритм приведем в операторной форме:

1) $i := 1$;

2) $h := h_0 + s\,i\,d$. Если $s\,\operatorname{sgn}\big(\varepsilon - \varepsilon(h)\big) < 0$, то $h = h_0 + s\,\sigma(i,h)\,d$ и переход к 5;

3) если $i = D - 1$, то $h_0 := h$, $d := h_0/D$, и переход к 1;

4) $i := i + 1$ и переход к 2;

5) выход.

2.1.4. Стандартный алгоритм коррекции шага. Он основан на апостериорной информации. Скорректированная величина h для текущего узла при известном значении $x = (x_1, \ldots, x_n)$ в нем и известном приближении h_0 вычисляется как $h = h_0 \left(\sqrt{\dfrac{1}{n}\sum\limits_{i=1}^{n} \left(\dfrac{\delta T(h_0)}{\Delta + \varepsilon \max\Big\{|x_i|, \big|T_i(h_0)\big|\Big\}}\right)^2}\right)^{1/(M+1)}$,

которая аналогична используемым в методах Рунге–Кутта [28] (M — порядок МРТ).

2.1.5. Градуировка. Она заключается в том, что для каждого порядка $p \in [M_{\min} : M_{\max}]$ (где, например, $M_{\min} = 5$, $M_{\max} = 60$) определяется процессорное время $t(p)$ вычисления всего набора коэффициентов Тейлора решения. Градуировка производится на начальном этапе работы программы до начала интегрирования системы на заданном промежутке, а ее результаты используются для выбора порядка и величины шага на первом шаге интегрирования и для корректировки порядка на следующих шагах.

2.1.6. Выбор величины шага и порядка на первом шаге. Начальное приближение h_0 для величины первого шага вычисляется по алгоритму 2.1.2. Затем при $p \in [M_{\min} : M_{\max}]$ вычисляются величины $V(p) = h(p)/t(p)$ — "пошаговые скорости", где $h(p)$ — величина шага, полученная МРТ порядка p с помощью алгоритма 2.1.3 и с использованием начального приближения h_0. В качестве порядка на

первом шаге выбирается величина M, такая, что $V(M) = \max\limits_{p\in\,[M_{\min},M_{\max}]} V(p)$, а в качестве величины этого шага принимается $h = h(M)$.

2.2. Алгоритм автоматического выбора шага. На каждом шаге (кроме первого) за начальное приближение h_0 принимается значение, полученное на предыдущем шаге интегрирования. Затем с помощью алгоритмов 2.1.2 и 2.1.4 вычисляются две величины шага h_1 и h_2. Из них выбирается максимальная — $h = \max\{h_1, h_2\}$, которая корректируется по алгоритму 2.1.3.

2.3. Алгоритм автоматического выбора порядка. Если на очередном шаге МРТ величина шага h изменилась в m (например, $m = 5$) раз по сравнению с величиной H (H изменяется после каждого изменения величины порядка M, а на первом шаге полагается равной величине первого шага), то производится корректировка порядка M. Предполагается, что известны коэффициенты Тейлора до порядка M. Для $p = M-1, \ldots, M_{\min}$ последовательно вычисляются длины шагов $h(p)$ и $V(p) = h(p)/t(p)$. Как только (и если) окажется, что $V(p) \geqslant V(M)$, новый порядок M полагается равным этому p. Если же не найдется таких $p = M-1, \ldots, M_{\min}$, то для $p = M+1, \ldots, M_{\max}$ последовательно вычисляются коэффициенты Тейлора порядка p (с учетом того, что коэффициенты порядка до p уже известны), соответствующий шаг $h(p)$ и скорость $V(p)$. Как только (и если) окажется, что $V(p) \geqslant V(M)$, новый порядок M полагается равным этому p. Если же не найдется таких $p = M+1, \ldots, M_{\max}$, то использовавшийся порядок не корректируется.

2.4. Алгоритм интегрирования на $[t_0, T]$**.** Ниже этот алгоритм представлен в операторной форме.

1. Выбор величины шага и порядка на первом шаге (п. 2.1.6) и присвоение этих значений переменным H и M (п. 2.3).

2. Вычисление в следующей точке t_k коэффициентов Тейлора по формулам (4), (5) и шага h (п. 2.2).

3. Если шаг h изменился более чем в m раз по сравнению с величиной H, то вычисляется новый порядок M (п. 2.3) и соответствующий шаг h (п. 2.2).

4. Если $t_k + h < T$, то $t_{k+1} = t_k + h$, вычисляется решение в точке t_{k+1} и переход к 2.

5. Вычисление решения в точке T (шаг: $h = T - t_k$), выход.

3. Примеры. В разделах 3.1 и 3.2 рассматриваются простейшее квадратичное уравнение, уравнения для эллиптических функций Якоби и уравнения Лоренца. В разделе 3.3 представлены результаты численного интегрирования уравнений движения внешних планет Солнечной системы в рамках классических моделей движения шести и двух тел на промежутке в двадцать четыре миллиона лет. В численных экспериментах для двух первых примеров и уравнений Лоренца использовалось представление данных типа quadruple precision, а для задачи двух тел и задачи о движении внешних планет — типа double precision. Как уже было сказано во введении, решение задачи Коши для каждого из примеров получалось при помощи четырех программ пошагового численного интегрирования: DOP853, ODEX, TIDES и TSMR. В программе TIDES пределы изменения порядка МРТ мы установили равными 5–60, как и в TSMR. Вычисления производились на компьютере с процессором Intel Xeon CPU E5440, 2.83 ГГц, ОЗУ 4 Гб и операционной системой CentOS 6, а в качестве компилятора использовался Intel Fortran 12. Отметим, что предельный порядок 60 был выбран нами исходя из возможностей имевшейся в нашем распоряжении вычислительной техники. В рассматриваемых примерах в программах TSMR и TIDES достигались порядки не более 39 и 37 соответственно. В других задачах и при более высоких требованиях на точность, вообще говоря, возможны и более высокие порядки. Результаты расчетов представлены в терминах фактической относительной погрешности решений и затраченного процессорного времени. По приводимым таблицам и графикам можно судить о сравнительной эффективности методов.

3.1. Квадратичная задача и уравнения для эллиптических функций Якоби.

3.1.1. Простейшая квадратичная задача (Simplest). Это задача Коши $\dfrac{dx}{dt} = x^2$, $x(0) = 1$, решением которой является функция $x(t) = \dfrac{1}{1-t}$. Подобные задачи в качестве тестовых дают представление об эффективности численного интегрирования тем или иным методом в окрестности особых точек решения.

3.1.2. Уравнения для эллиптических функций Якоби (Jacobi). Это задача Коши $\dfrac{dx_1}{dt} = x_2 x_3$, $\dfrac{dx_2}{dt} = -x_1 x_3$, $\dfrac{dx_3}{dt} = -0.5 x_1 x_2$, $x_1(0) = 0$, $x_2(0) = 1$, $x_3(0) = 1$, решение которой состоит из эллиптических функций Якоби: $x_1(t) = \mathrm{sn}\,(t)$, $x_2(t) = \mathrm{cn}\,(t)$, $x_3(t) = \mathrm{dn}\,(t)$. Эта, популярная в качестве тестовой, задача аналогична по структуре задаче Эйлера о движении твердого тела около неподвижной точки при нулевых моментах действующих на него сил.

Результаты численных экспериментов для этих задач представлены на рис. 1 и 2. Дадим некоторые

ε	Метод	SIMPLEST, $[0, 1–10^{-5}]$		SIMPLEST, $[0, 1–10^{-9}]$	
		ge	t^{CPU}	ge	t^{CPU}
10^{-10}	DOP853	1.3×10^{-6}	7.0×10^{-5}	1.1×10^{-1}	1.1×10^{-4}
	ODEX	9.1×10^{-7}	9.7×10^{-5}	8.3×10^{-2}	1.5×10^{-4}
	TIDES	8.7×10^{-2}	3.4×10^{-5}	1.0	3.0×10^{-5}
	TSMR	3.4×10^{-7}	1.0×10^{-4}	5.2×10^{-2}	1.0×10^{-4}
	TSMR_A	9.6×10^{-8}	5.0×10^{-5}	9.9×10^{-3}	1.5×10^{-4}
10^{-20}	DOP853	8.4×10^{-17}	2.1×10^{-2}	8.4×10^{-12}	3.5×10^{-2}
	ODEX	3.2×10^{-17}	7.3×10^{-3}	3.2×10^{-12}	1.1×10^{-2}
	TIDES	4.9×10^{-12}	2.4×10^{-3}	4.0×10^{-2}	3.8×10^{-3}
	TSMR	2.6×10^{-17}	5.2×10^{-3}	3.8×10^{-12}	7.7×10^{-3}
	TSMR_A	1.6×10^{-18}	4.0×10^{-3}	1.3×10^{-13}	9.0×10^{-3}
10^{-30}	DOP853	6.5×10^{-25}	3.6×10^{-1}	6.5×10^{-20}	1.5×10^{-1}
	ODEX	2.7×10^{-26}	2.4×10^{1}	1.9×10^{-5}	4.1×10^{-1}
	TIDES	2.1×10^{-22}	4.7×10^{-3}	2.5×10^{-12}	5.7×10^{-3}
	TSMR	2.7×10^{-27}	1.0×10^{-2}	2.7×10^{-22}	1.3×10^{-2}
	TSMR_A	1.8×10^{-29}	9.0×10^{-3}	5.3×10^{-25}	1.3×10^{-2}

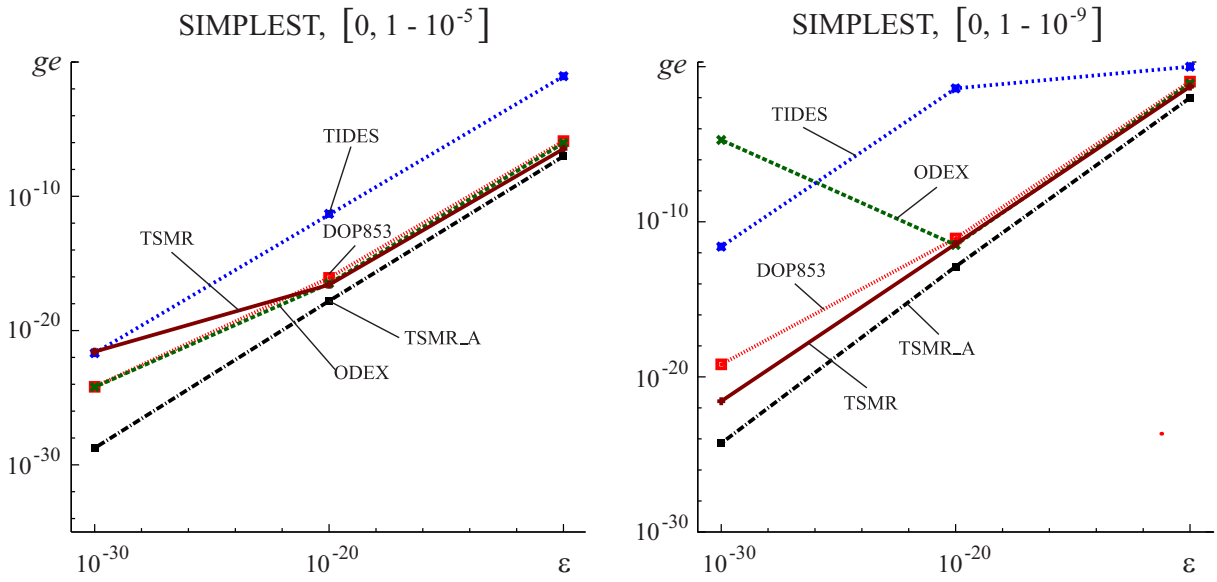

Рис. 1. К задаче SIMPLEST: ε — требуемая относительная погрешность, ge — максимальная из относительных погрешностей координат в конце промежутка, t^{CPU} — процессорное время (сек.)

пояснения к ним:

— справа от названия задачи (SIMPLEST, JACOBI) приводится временной промежуток, на котором интегрировалась эта задача;

— значение функций Якоби на промежутке $[0, 100K + 1]$ (K — период этих функций) с необходимой точностью вычислялось в программе Wolfram Mathematica [31];

— в первой колонке приводится требуемая относительная погрешность $\varepsilon = 10^{-10}, \ldots, 10^{-30}$ — величина, которую не должна превосходить максимальная по модулю из всех относительных погрешностей координат решения в конце промежутка интегрирования;

— во второй колонке метод TSMR_A — это тот же метод TSMR, но с использованием масштабирующих множителей (п. 1.3.2), т.е. предполагается, что пользователь задал подпрограмму вычисления масштабирующих множителей (п. 2.1.2 и статья [32], в которой приведены формулы вычисления масштабирующих множителей для рассматриваемых задач);

ε	Метод	JACOBI, $[0, 100K+1]$	
		ge	t^{CPU}
10^{-10}	DOP853	3.6×10^{-8}	4.3×10^{-4}
	ODEX	2.2×10^{-8}	7.5×10^{-4}
	TIDES	3.0×10^{-8}	7.1×10^{-4}
	TSMR	5.2×10^{-9}	1.6×10^{-3}
	TSMR_A	1.3×10^{-13}	2.0×10^{-3}
10^{-20}	DOP853	2.4×10^{-19}	3.7×10^{-1}
	ODEX	1.0×10^{-17}	9.0×10^{-2}
	TIDES	8.2×10^{-20}	6.2×10^{-2}
	TSMR	2.3×10^{-25}	1.6×10^{-1}
	TSMR_A	9.3×10^{-30}	2.0×10^{-1}
10^{-30}	DOP853	1.4×10^{-27}	6.4
	ODEX	1.5×10^{-29}	2.9×10^{2}
	TIDES	7.2×10^{-30}	1.5×10^{-1}
	TSMR	5.9×10^{-30}	2.0×10^{-1}
	TSMR_A	9.0×10^{-32}	3.2×10^{-1}

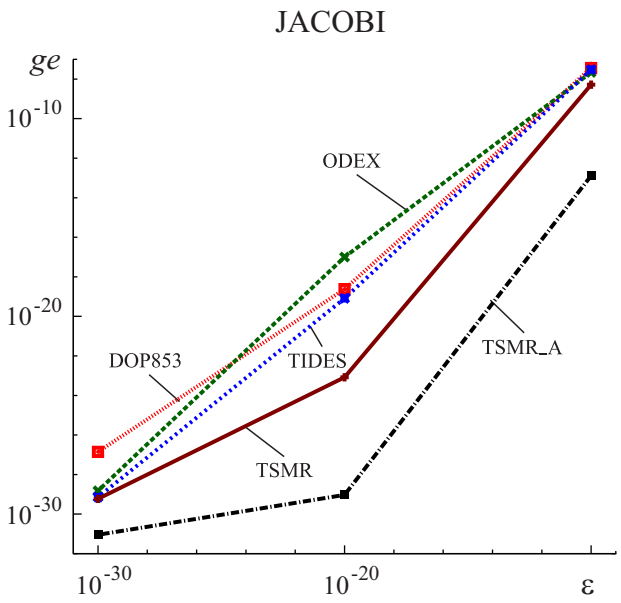

Рис. 2. К задаче JACOBI: ε — требуемая относительная погрешность, ge — максимальная из относительных погрешностей координат в конце промежутка, t^{CPU} — процессорное время (сек.)

— в третьей и пятой колонках ge означает максимальную по модулю из всех относительных погрешностей координат решения в конце промежутка интегрирования;

— в четвертой и шестой колонках t^{CPU} означает затраченное процессорное время.

Представленные таблично и графически результаты понять несложно, они не требуют каких-либо специальных пояснений.

3.2. Уравнения Лоренца (Lorenz). Решение задачи Коши $\dot{x} = -10x + 10y$, $\dot{y} = -xz + 28x - y$, $\dot{z} = xy - 8z/3$, $x(0) = -13.763610682134200525014401054362$, $y(0) = -19.578751942451795538838041446010$, $z(0) = 27$ — периодическое с периодом $T = 1.558652210716174727567870209212 7$. В работе [9] (откуда мы и взяли эти данные) приведены результаты численных экспериментов для этой популярной задачи. В частности, там продемонстрировано, что метод DOP853 при вычислениях с данными типа double precision уже на промежутке $[0, 50T]$ дает совершенно неудовлетворительные результаты и что программа TIDES при вычислениях с гораздо большим количеством значащих цифр (200) дает достаточно точные результаты на промежутке $[0, 300T]$ (в программе TIDES возможны вычисления с любым предопределенным количеством значащих цифр). В наших численных экспериментах для задачи Лоренца мы провели вычисления с данными типа quadruple precision на промежутках времени $[0, T], \ldots, [0, 50T]$, с тем чтобы сравнение методов по их эффективности было корректным.

В таблице на рис. 3 для использованных методов приведены данные об относительной погрешности и затраченном процессорном времени решения задачи Лоренца. Здесь приняты те же обозначения, что и на рис. 1 и 2, причем для TSMR_A использовался алгоритм вычисления масштабирующих множителей, который мы ниже рассмотрим. Почти до самого конца интервала программа TSMR_A показала бóльшую точность, но уступала TSMR по затратам процессорного времени, но к сорок девятому периоду все программы сравнялись по точности (вернее — "по неточности").

Вернемся к масштабирующим множителям. В соответствии с (8) для задачи Лоренца получаем

$$s_1 = 10 + 10\alpha_2/\alpha_1 = a + b\alpha_2/\alpha_1, \quad s_3 = 8/3 + \alpha_1\alpha_2/\alpha_3 = e + f\alpha_1\alpha_2/\alpha_3,$$
$$s_2 = 1 + 28\alpha_1/\alpha_2 + \alpha_1\alpha_3/\alpha_2 = 1 + c\alpha_1/\alpha_2 + d\alpha_1\alpha_3/\alpha_2,$$

и приходим к задаче: при a, b, c, d, e, $f > 0$, $s_1 = a + b\alpha_2/\alpha_1$, $s_2 = 1 + c\alpha_1/\alpha_2 + d\alpha_1\alpha_3/\alpha_2$, $s_3 = e + f\alpha_1\alpha_2/\alpha_3$, найти $\alpha_1, \alpha_2, \alpha_3 \geqslant 1$, доставляющие минимум величине $\max\{s_1, s_2, s_3\}$.

Вводя замену $u = \alpha_1/\alpha_2$, $v = \alpha_1\alpha_3/\alpha_2$, $w = \alpha_1\alpha_2/\alpha_3$; $\alpha_1 \geqslant 1$, $\alpha_2 \geqslant 1$, $\alpha_3 \geqslant 1$, $\alpha_1 = \sqrt{vw}$, $\alpha_2 = u\sqrt{vw}$, $\alpha_3 = vu$; $u, v, w > 0$, $vw \geqslant 1$, $u^2vw \geqslant 1$, $vu \geqslant 1$, перейдем от исходной задачи к задаче нахождения точки

Метод	$[0, A]$	ge	t^{CPU}
DOP853	$A = 20T$	1.3×10^{-17}	11
ODEX		5.5×10^{-20}	950
TIDES		6.7×10^{-20}	0.31
TSMR		1.2×10^{-20}	0.22
TSMR_A		3.4×10^{-21}	10
DOP853	$A = 30T$	6.8×10^{-11}	17
ODEX		3.0×10^{-13}	1420
TIDES		3.6×10^{-13}	0.46
TSMR		1.2×10^{-12}	0.30
TSMR_A		1.2×10^{-13}	16
DOP853	$A = 40T$	3.7×10^{-4}	22
ODEX		4.2×10^{-5}	1910
TIDES		2.0×10^{-6}	0.62
TSMR		6.6×10^{-6}	0.41
TSMR_A		5.4×10^{-7}	20
DOP853	$A = 46T$	4.0×10^{-1}	26
ODEX		8.1×10^{-2}	2180
TIDES		2.2×10^{-2}	0.71
TSMR		7.8×10^{-2}	0.45
TSMR_A		3.2×10^{-3}	21
DOP853	$A = 48T$	9.2×10^{-1}	27
ODEX		5.8×10^{-1}	2280
TIDES		1.1×10^{-1}	0.72
TSMR		7.7×10^{-1}	0.47
TSMR_A		2.1×10^{-1}	22
DOP853	$A = 49T$	1.0	27
ODEX		2.0	2330
TIDES		1.9	0.75
TSMR		2.0	0.49
TSMR_A		8.1×10^{-1}	23

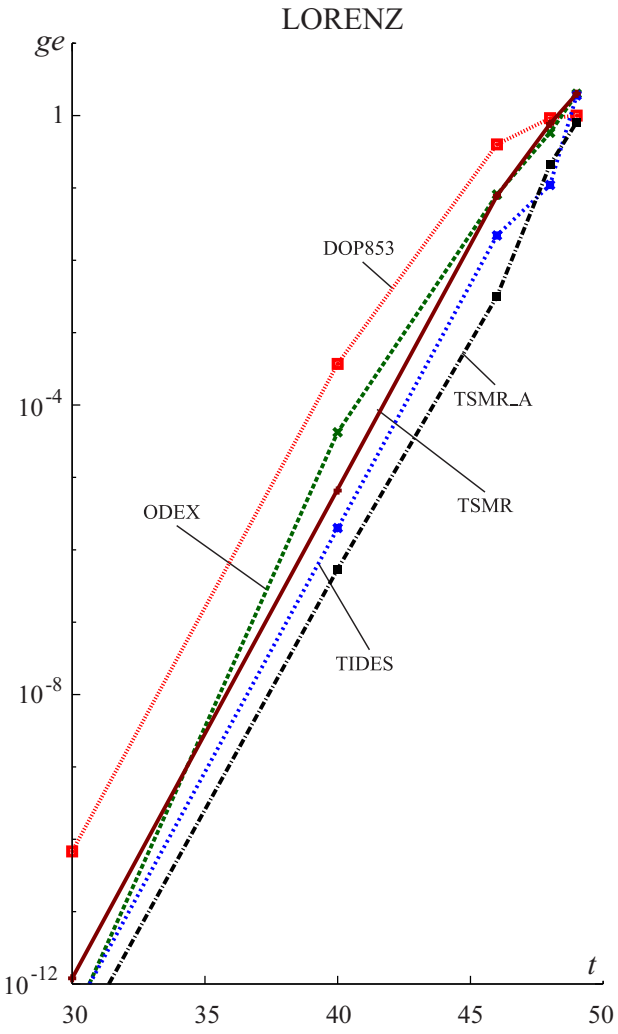

Рис. 3. К интегрированию задачи Лоренца: ge — максимальная из относительных погрешностей координат в конце промежутка, t^{CPU} — процессорное время (сек.), T — период решения задачи Лоренца. Результаты в таблице получены при требуемой относительной погрешности $\varepsilon = 10^{-30}$

(u, v, w) (она может быть не единственной), доставляющей минимум величине

$$s(u, v, w) = \max\{s_1 = a + bu,\ s_2 = 1 + c/u + dv,\ s_3 = e + fw\}$$

при $(u, v, w) \in D = \Big\{(u, v, w)\Big|u, v, w > 0,\ vw \geqslant 1,\ u^2vw \geqslant 1,\ vu \geqslant 1\Big\}$. Если $s_* = s(u_1, v_1, w_1)$ — минимальное значение величины $s(u, v, w)$ при некотором наборе $(u_1, v_1, w_1) \in D$, то оно достигается и при некотором наборе $(u, v, w) \in D$, таком, что $s_* = s_1 = s_2 = s_3$. Это следует из того, что s_1, s_2, s_3 строго возрастают по u, v, w соответственно и не возрастают по остальным двум переменным, и из того, что $(u_1, v_1, w_1) \in D$, $u \geqslant u_1,\ v \geqslant v_1,\ w \geqslant w_1 \Rightarrow (u, v, w) \in D$. Действительно, имея это в виду, выполнение условий $s_* = s_1 = s_2 = s_3$ без увеличения величины s_* можно обеспечить, увеличив w или v при $s_* = s_1 = s_2 > s_3$ и $s_* = s_1 = s_3 > s_2$ соответственно, увеличив w и v при $s_* = s_1 > s_2,\ s_* > s_3$, увеличив u и v при условии

$s_* = s_2 = s_3 > s_1$ или $s_* = s_3 > s_2,\ s_* > s_1$, и увеличив u, v, w при условии $s_* = s_2 > s_1,\ s_* > s_3$. Выражая эти искомые значения u, v, w через s_*, получаем $u = (s_* - a)/b,\ v = (bu - c/u + a - 1)/d,\ w = (bu + a - e)/f$ и приходим к задаче нахождения минимального s_*, удовлетворяющего неравенствам $u, v, w > 0,\ vw \geqslant 1,\ u^2vw \geqslant 1,\ vu \geqslant 1$, или, иначе, к нахождению минимального $u = u_*$, удовлетворяющего неравенствам

$$u > 0, \quad bu - A/u + a - 1 > 0, \quad bu + a - e > 0, \quad (bu - c/u + a - 1)u \geqslant d, \tag{9}$$

$$(bu - c/u + a - 1)(bu + a - e) \geqslant df, \quad u^2(bu - c/u + a - 1)(bu + a - e) \geqslant df. \tag{10}$$

Второе неравенство в (9) можно не рассматривать, так как оно следует из первого и последнего неравенств этой группы. Тогда вместо (9) получаем

$$u > 0, \quad u > (e - a)/b, \quad u \geqslant \frac{1}{2b}\left(1 - a + \sqrt{(1 - a)^2 + 4b(c + d)}\right),$$

т.е. неравенства (9) выполняются тогда и только тогда, когда

$$u \geqslant u' = \max\left\{\frac{e - a}{b}, \frac{1}{2b}\left(1 - a + \sqrt{(1 - a)^2 + 4b(c + d)}\right)\right\}. \tag{11}$$

Первое из неравенств (10) следует из второго при $0 < u \leqslant 1$, а второе из первого — при $u \geqslant 1$, поэтому вместо (10) можно рассмотреть две системы неравенств:

$$u \geqslant 1, \quad \phi(u) = (bu - c/u + a - 1)(bu + a - e) - df \geqslant 0, \tag{12}$$

$$0 < u \leqslant 1, \quad \psi(u) = u^2(bu - c/u + a - 1)(bu + a - e) - df \geqslant 0. \tag{13}$$

Если выполнены неравенства (9) (а значит, и (11)), то функции $\phi(u)$, $\psi(u)$ строго возрастающие. Если u'' и u''' — минимальные значения u, удовлетворяющие (12) и (13) соответственно, то $u_* = \max\{u', u'', u'''\}$. Сказанное позволяет сформулировать алгоритм нахождения оптимальных значений u, v, w, α_1, α_2, α_3, s по a, b, c, d, e, $f > 0$ (с использованием функций $\phi(u)$, $\psi(u)$, см. (12), (13)).

Алгоритм нахождения α_1, α_2, α_3:

1) $u' = \max\left\{\frac{e - a}{b}, \frac{1}{2b}\left(1 - a + \sqrt{(1 - a)^2 + 4b(c + d)}\right)\right\}$;

2) $u' \geqslant 1 \Rightarrow u_* = \begin{cases} u', & \phi(u') \geqslant 0, \\ u'', & \phi(u') < 0, \quad u'' \in (u', +\infty) \text{ — единственный корень } \phi(u); \end{cases}$

3) $u' < 1 \Rightarrow u_* = \begin{cases} u', & \psi(u') \geqslant 0, \\ u'', & \psi(u') < 0, \quad \psi(1) \geqslant 0, \quad u'' \in (1, +\infty) \text{ — единственный корень } \phi(u), \\ u''', & \psi(u') < 0, \quad \psi(1) < 0, \quad u''' \in (u', 1] \text{ — единственный корень } \psi(u); \end{cases}$

4) выходные данные: $u = u_*,\ v = (bu - c/u + a - 1)/d,\ w = (bu + a - e)/f,\ \alpha_1 = \sqrt{vw}\,,$
$\alpha_2 = u\sqrt{vw}\,,\ \alpha_3 = vu,\ s = bu_* + a.$

3.3. Орбитальное движение внешних планет и задача двух тел. Уравнения движения внешних планет Солнечной системы в гелиоцентрических координатах имеют вид [33]

$$\ddot{g}_{ij} = -k^2(m_0 + m_i)\frac{g_{ij}}{r_{0i}^3} + k^2 \sum_{\substack{s \in [1:5], \\ s \neq i}} m_s\left[\frac{g_{sj} - g_{ij}}{r_{si}^3} - \frac{g_{sj}}{r_{0s}^3}\right],$$

где $r_{si}^2 = \sum\limits_{j \in\ [1:3]} (g_{ij} - g_{sj})^2,\ r_{si} > 0,\ i \in\ [1:5],\ s \in\ [0:5],\ s \neq i,\ j = 1, 2, 3,\ k$ — постоянная Гаусса. Рассмотрим их в полиномиальной форме [1]

$$\dot{p}_{ij} = -k^2(m_0 + m_i)g_{ij}d_{0i}^3 + k^2 \sum_{\substack{s \in [1:5], \\ s \neq i}} m_s\left[(g_{sj} - g_{ij})\,d_{si}^3 - g_{sj}d_{0s}^3\right], \quad \dot{g}_{ij} = p_{ij},$$

$$\dot{d}_{si} = -d_{si}^3 \sum_{j \in\ [1:3]} (g_{ij} - g_{sj})(p_{ij} - p_{sj}), \quad i \in\ [1:5], \quad s \in\ [0:5], \quad s < i,$$

где $g_{i1} = x_i,\ g_{i2} = y_i,\ g_{i3} = z_i,\ d_{si} = 1/r_{si}$. Хотя уравнения задачи двух тел и аналогичны, выпишем их отдельно:

$$\dot g_j = p_j, \quad \dot p_j = -k^2(m_0 + m_1)g_j/r^3, \quad r^2 = g_1^2 + g_2^2 + g_3^2, \quad j = 1, 2, 3;$$
$$\dot g_j = p_j, \quad \dot p_j = -k^2(m_0 + m_1)g_j d^3, \quad \dot d = -d^3(g_1p_1 + g_2p_2 + g_3p_3), \quad j = 1, 2, 3, \quad d = 1/r.$$

Гелиоцентрические координаты и скорости планет представлены в табл. 1. Мы взяли их из статьи [34], а их массы — на сайте NASA Jet Propulsion Laboratory [35].

Таблица 1

Координаты (AU), скорости (AU/d) и массы внешних планет

Планета	$g_{i1},\ g_{i2},\ g_{i3}$	$p_{i1},\ p_{i2},\ p_{i3}$
Юпитер ($m_\odot/m_{\text{Jupiter}} = 1047.3486$)	$0.3648\ 4424\ 2316\ 710 \times 10^{1}$	$0.5145\ 8739\ 1602\ 440 \times 10^{-2}$
	$-0.3188\ 5628\ 5618\ 430 \times 10^{1}$	$0.5377\ 0578\ 5677\ 460 \times 10^{-2}$
	$-0.1457\ 0594\ 1385\ 510 \times 10^{1}$	$0.2180\ 9890\ 3370\ 320 \times 10^{-2}$
Сатурн ($m_\odot/m_{\text{Saturn}} = 3497.898$)	$0.8608\ 1200\ 8268\ 000 \times 10^{-1}$	$-0.5877\ 1844\ 0129\ 930 \times 10^{-2}$
	$0.8332\ 3915\ 0335\ 020 \times 10^{1}$	$-0.4807\ 7953\ 8090\ 000 \times 10^{-4}$
	$0.3441\ 6852\ 4680\ 770 \times 10^{1}$	$0.0234\ 4003\ 6257\ 520 \times 10^{-2}$
Уран ($m_\odot/m_{\text{Uranus}} = 22902.98$)	$-0.1689\ 4580\ 3211\ 040 \times 10^{2}$	$0.1537\ 1000\ 2472\ 550 \times 10^{-2}$
	$-0.6802\ 7902\ 7446\ 500 \times 10^{1}$	$-0.3460\ 0609\ 5920\ 100 \times 10^{-2}$
	$-0.2742\ 0152\ 0254\ 400 \times 10^{-1}$	$-0.1537\ 8414\ 5913\ 290 \times 10^{-2}$
Нептун ($m_\odot/m_{\text{Neptune}} = 19412.24$)	$-0.1196\ 5863\ 9802\ 400 \times 10^{2}$	$0.2867\ 5086\ 9894\ 660 \times 10^{-2}$
	$-0.2587\ 3934\ 3955\ 110 \times 10^{2}$	$-0.1099\ 5281\ 7836\ 360 \times 10^{-2}$
	$-0.1029\ 7822\ 0281\ 190 \times 10^{2}$	$-0.5224\ 5552\ 2316\ 600 \times 10^{-3}$
Плутон ($m_\odot/m_{\text{Pluto}} = 1.35 \times 10^{8}$)	$-0.2962\ 8255\ 3361\ 990 \times 10^{2}$	$0.7472\ 8976\ 7668\ 000 \times 10^{-3}$
	$-0.5542\ 2379\ 5055\ 800 \times 10^{1}$	$-0.3094\ 8199\ 5153\ 220 \times 10^{-2}$
	$0.7229\ 0409\ 9388\ 100 \times 10^{1}$	$-0.1202\ 5228\ 5627\ 570 \times 10^{-2}$

В численных экспериментах с этими уравнениями и данными мы рассматриваем задачу о движении пяти внешних планет вокруг Солнца (т.е. задачу Солнце–Юпитер–Сатурн–Уран–Нептун–Плутон) и пять задач двух тел: Солнце–Юпитер, Солнце–Сатурн, Солнце–Уран, Солнце–Нептун, Солнце–Плутон. Во всех этих задачах полагаем $m_\odot = 1$, а начальные координаты и скорости берем из табл. 1. Для интегрирования, кроме нашей программы TSMR, используем программы DOP853, ODEX, TIDES. Для трех последних программ используем обозначения DOP853_P, ODEX_P, TIDES_P в случае, когда решаются уравнения рассматриваемых задач в полиномиальной форме (напомним, что программа TSMR ориентирована на решение полиномиальных систем, см. введение).

Задачу о движении внешних планет интегрируем всеми указанными методами на временны́х промежутках вида $[0, T]$ при $T = 1$ млн. лет, $\dots$, 24 млн. лет по схеме “туда–обратно”, т.е. на каждом из промежутков сначала интегрируем уравнения от точки 0 до точки T, а затем, принимая полученные данные за начальные, интегрируем уравнения от точки T до точки 0. Результаты интегрирования представлены на рис. 4а, где $ge_{\rightleftarrows}$ — максимальная из величин модулей относительных погрешностей координат и скоростей планет при возврате в точку 0, а T — длина промежутка интегрирования.

Если предположить, что схема интегрирования “туда–обратно” дает верные оценки для относительных погрешностей, то можно сделать вывод, что чем меньше в точке T значение $ge_{\rightleftarrows}$ на графике, тем более точным оказалось вычисление этой величины в этой точке при помощи соответствующей программы. Если рассуждать таким образом, то можно расположить использовавшиеся программы по убыванию обеспечиваемой ими точности интегрирования уравнений движения внешних планет.

К сожалению, при отсутствии практически адекватных строгих оценок глобальных погрешностей в рассмотренной задаче мы можем только предполагать, что примененная выше схема интегрирования “туда–обратно” в какой-то степени адекватно отражает поведение относительных погрешностей. Чтобы укрепиться в этом мнении и получить дополнительные аргументы, мы рассмотрели уравнения движения

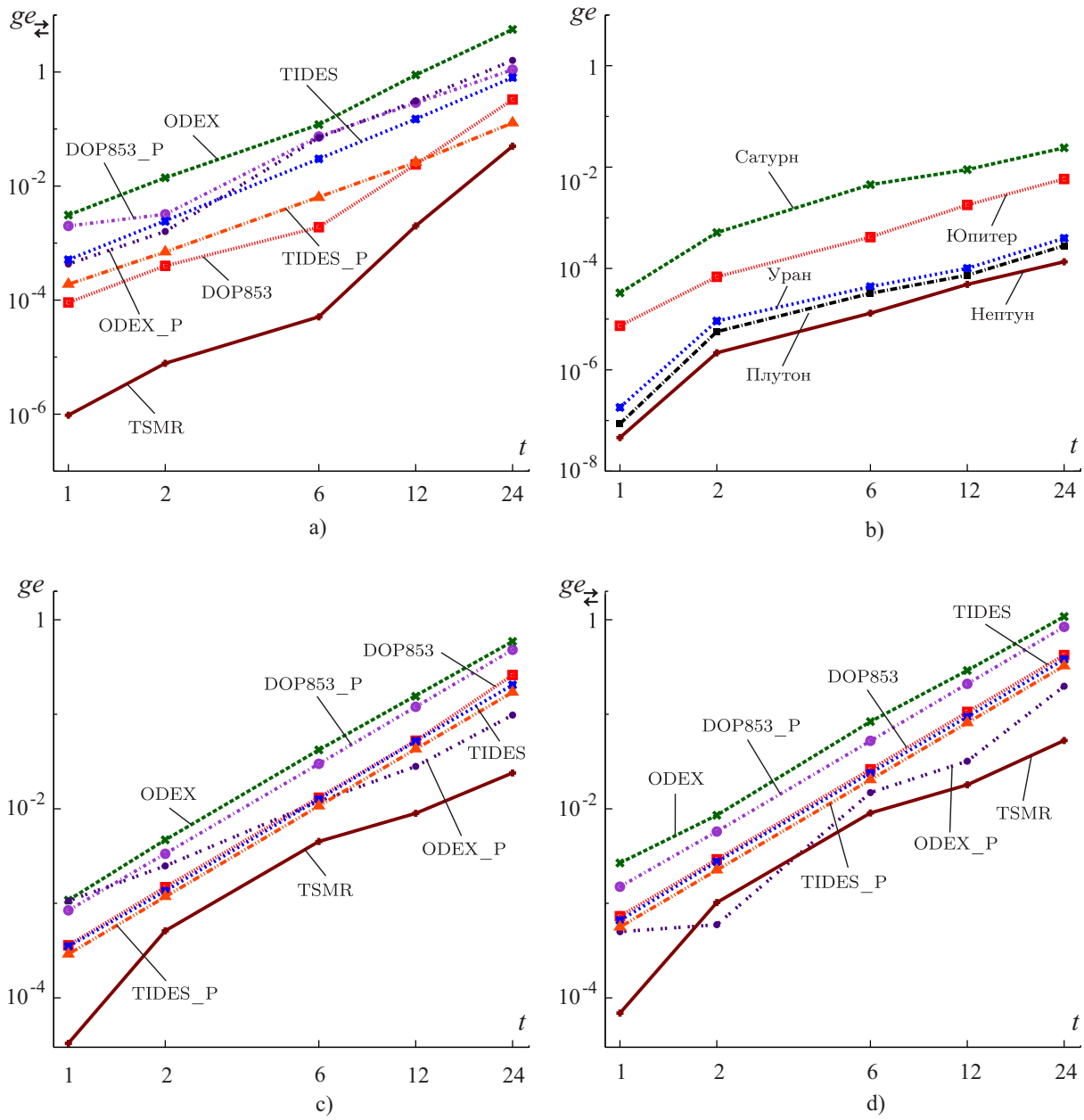

Рис. 4. Графики погрешностей интегрирования на $[0, T]$ (T — в млн. лет): a) для пяти внешних планет, b) для пяти задач двух тел программой TSMR, c) и d) для задачи Солнце–Сатурн

каждой из пяти внешних планет (при своих массах и начальных данных, взятых из табл. 1), проинтегрировав их на том же промежутке в двадцать четыре миллиона лет только программой TSMR по схеме “туда” (рис. 4b).

Отметим, что обозначения на графиках a и b аналогичны, но у функции ge на графике b отсутствует значок $\rightleftarrows$, так как ее аргументом является t, а не T. Так как решения рассмотренных задач двух тел периодические и мы могли вычислить значения относительных погрешностей с необходимой точностью (мы вычисляли эти значения в кратных периоду планеты точках, а сами эти точки предвычислили с достаточной точностью в программе Wolfram Mathematica), то эти графики правильно отражают поведение величин ge. Анализ графиков дает основание предположить, что поведение исследуемых относительных погрешностей в задаче о движении внешних планет более всего определяется динамикой Сатурна.

Так мы пришли к задаче интегрирования уравнений движения Сатурна при помощи всех семи вариантов программ, которые мы использовали в задаче о движении пяти внешних планет. Результаты представлены на рис. 4: c) схема интегрирования “туда”; d) схема интегрирования “туда–обратно”.

По графикам для функций ge, $ge_{\rightleftarrows}$ определенно можно сделать вывод, что в задаче Солнце–Сатурн программа TSMR дает наиболее близкие к истинным результаты.

3.3.1. Анализ результатов интегрирования уравнений движения планет различными программами. По приведенным выше графикам на рис. 4, отражающим погрешность численного интегрирования уравнений движения внешних планет, при помощи различных программ составлена табл. 2, содержащая рейтинги каждой программы по точности вычисления величин ge, $ge_{\rightleftarrows}$ и затраченное этой программой на интегрирование уравнений процессорное время.

Таблица 2

Рейтинг по относительной погрешности и процессорное время программ при решении планетных задач

Задача	Метод	Рейтинг		t^{CPU}
		ge	$ge_{\rightleftarrows}$	(сек.)
Задача Солнце–Сатурн, “туда”	DOP853	3–5		83
	DOP853_P	6		77
	ODEX	7		52
	ODEX_P	2–5		52
	TIDES	2–3		42
	TIDES_P	3–4		53
	TSMR	1		49
Солнце–Сатурн, “туда-обратно”	DOP853		5	167
	DOP853_P		6	154
	ODEX		7	105
	ODEX_P		1–2	104
	TIDES		4	85
	TIDES_P		3	106
	TSMR		1–2	98
Солнце–Юпитер–Сатурн–Уран–Нептун–Плутон, “туда-обратно”	DOP853		2–3	3420
	DOP853_P		5–6	5580
	ODEX		7	1920
	ODEX_P		4–5	4020
	TIDES		4–5	7500
	TIDES_P		2–3	9360
	TSMR		1	8340

Отметим, что приведенные на графиках и в табл. 2 данные по погрешностям для задачи двух тел Солнце–Сатурн являются достоверными, а для задачи шести тел — скорее истинными, чем нет. Если считать все же все эти данные истинными, то можно отметить следующее относительно эффективности программы TSMR в рассмотренных задачах о движении планет:

— программа TSMR показала себя лучшей по точности во всех рассмотренных задачах;

— по затраченному процессорному времени она была второй в задачах двух тел, но оказалась шестой — в задаче шести тел (вообще, все три варианта МРТ в задаче двух тел оказались наиболее быстрыми, а в задаче шести тел самыми медленными).

Полученные результаты можно считать естественными: при увеличении сложности правой части и размерности затраты процессорного времени для методов рядов Тейлора растут быстрее, чем у методов Рунге–Кутта не очень высокого порядка.

4. Заключение. Основные результаты статьи состоят в следующем. В разделе 2 изложен алгоритм, реализующий новый вариант явного метода рядов Тейлора пошагового решения нежестких полиномиальных дифференциальных уравнений [1]. Представленные в разделе 3 результаты численных экспериментов, проведенных при помощи программ численного интегрирования, основанных на явных методах

Дормана–Принса, Грегга–Булирша–Штера, методе рядов Тейлора с использованием алгоритма автоматического дифференцирования (DOP853, ODEX, TIDES) и предложенной в настоящей работе программы TSMR, показали надежность и конкурентоспособность последней при решении как стандартных тестовых примеров, так и сложных задач динамики. Программу TSMR с инструкцией и примерами ее применения можно получить по адресу [29]. Она реализует явный метод рядов Тейлора и ориентирована на решение нежестких задач динамики, представимых дифференциальными уравнениями в полиномиальной форме. Ее рекомендуется применять при высоких требованиях по относительной погрешности.

СПИСОК ЛИТЕРАТУРЫ

1. *Бабаджанянц Л.К.* Метод рядов Тейлора // Вестн. Санкт-Петербургского ун-та. Сер. 10. 2010. № 3. 13–29.
2. *Бабаджанянц Л.К.* Метод дополнительных переменных // Вестн. Санкт-Петербургского ун-та. Сер. 10. 2010. № 1. 3–11.
3. *Бабаджанянц Л.К., Брэгман К.М.* Алгоритм метода дополнительных переменных // Вестн. Санкт-Петербургского ун-та. Сер. 10. 2012. № 2. 3–12.
4. *Rall L.B.* Automatic differentiation: techniques and applications // Lecture Notes in Computer Science. Vol. 120. Berlin: Springer-Verlag, 1981.
5. *Corliss G.F., Chang Y.F.* Solving ordinary differential equations using Taylor series // ACM Trans. on Math. Software. 1982. **8**. 114–144.
6. *Berz M., Bischof C., Corliss G.F., Griewank A.* Computational differentiation: techniques, applications, and tools. Philadelphia: SIAM, 1996.
7. *Lara M., Elipe A., Palacios M.* Automatic programming of recurrent power series // Math. Comput. Simul. 1999. **49**. 351–362.
8. *Griewank A.* Evaluating derivatives. Philadelphia: SIAM, 2000.
9. *Abad A., Barrio R., Blesa F., Rodriguez M.* Breaking the limits: the Taylor series method // Appl. Math. and Computation. 2011. **217**, N 20. 7940–7954.
10. *Rodriguez M., Barrio R.* Reducing rounding errors and achieving Brouwer's law with Taylor series method // Appl. Numer. Math. 2012. **62**, N 8. 1014–1024.
11. *Parker G.E., Sochacki J.S.* Implementing the Picard iteration // Neural, Parallel and Scientific Computation. 1996. **4**. 97–112.
12. *Parker G.E., Sochacki J.S.* A Picard–McLaurin theorem for initial value PDE's // Abstract and Appl. Analysis. 2000. **5**. 47–63.
13. *Pruett C.D., Rudmin J.W., Lacy J.M.* An adaptive N-body algorithm of optimal order // J. of Comput. Physics. 2003. **187**. 298–317.
14. *Carothers D.C., Parker G.E., Sochacki J.S., Warne P.G.* Some properties of solutions to polynomial systems of differential equations // Electron. J. Diff. Eqns. 2005. N 40. 1–17.
15. *Miletics E., Molnárka G.* Taylor series method with numerical derivatives for initial value problems // J. Comput. Methods in Sciences and Engineering. 2004. **4**, N 1–2. 105–114.
16. *Molnárka G., Miletics E.* Implicit extension of Taylor series method with numerical derivatives for initial value problems // Computers & Mathematics with Applications. 2005. **50**, N 7. 1167–1177.
17. *Nedialkov N.S., Jackson K.R., Corliss G.F.* Validated solutions of initial value problems for ordinary differential equations // Appl. Math. Comput. 1999. **105**. 21–68.
18. *Hoefkens J., Berz M., Makino K.* Computing validated solutions of implicit differential Equations // Adv. Comput. Math. 2003. **19**. 231–253.
19. *Chang Y.F., Corliss G.* ATOMFT: solving ODEs and DAEs using Taylor series // Comput. Math. Appl. 1994. **28**, N 10–12. 209–233.
20. *Berz M.* Cosy infinity version 8 reference manual. Technical Report MSUCL-1088. National Superconducting Cyclotron Lab., Michigan State University, East Lansing. Mich., 2003.
21. *Makino K., Berz M.* Taylor models and other validated functional inclusion methods // Int. J. Pure Appl. Math. 2003. **6**, N 3. 239–316.
22. *Jorba A., Zou M.* A software package for the numerical integration of ODEs by means of high-order Taylor methods // Exp. Math. 2005. **14**, N 1. 99–117.
23. *Nedialkov N.S., Pryce J.D.* Solving differential-algebraic equations by Taylor series. I. Computing Taylor coefficients // BIT. 2005. **45**, N 3. 561–591.
24. *Nedialkov N.S., Pryce J.D.* Solving differential-algebraic equations by Taylor series. II. Computing the system Jacobian // BIT. 2007. **47**, N 1. 121–135.
25. *Nedialkov N.S., Pryce J.D.* Solving differential algebraic equations by Taylor series. III. The DAETS code // J. Numer. Anal. Ind. Appl. Math. 2008. **3**, N 1–2. 61–80.
26. TIDES webpage URL: gme.unizar.es/software/tides.
27. *Hairer E.* webpage URL: www.unige.ch/~hairer/.
28. *Hairer E., Wanner G., Norsett S.P.* Solving ordinary differential equations: nonstiff problems. Berlin: Springer, 2009.

29. *Babadzanjanz L.K.* webpage URL: www.apmath.spbu.ru/ru/staff/babadzhanyants/.
30. *Арушанян О.Б., Залеткин С.Ф.* Численное решение обыкновенных дифференциальных уравнений на Фортране. М.: Изд-во Моск. ун-та, 1990.
31. Wolfram Mathematica webpage URL: www.reference.wolfram.com/mathematica/guide/Mathematica.html.
32. *Babadzhanjanz L.K., Sarkissian D.R.* Taylor series method for dynamic systems with control: convergence and error estimates // J. Math. Sci. 2006. **139**, N 6. 7025–7046.
33. *Абалакин В.К., Аксенов Е.П., Гребеников Е.А., Демин В.Г., Рябов Ю.А.* Справочное руководство по небесной механике и астродинамике. М.: Наука, 1976.
34. *Oesterwinter C., Cohen C.J.* New orbital elements for Moon and planets // Celestial Mech. 1972. **5**, N 3. 317–395.
35. NASA Jet Propulsion Laboratory URL: ssd.jpl.nasa.gov/?constants.

Поступила в редакцию
29.05.2012