УДК 519.622

ПРОСТЫЕ ЯВНЫЕ МЕТОДЫ ЧИСЛЕННОГО РЕШЕНИЯ ЖЕСТКИХ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Л. М. Скворцов[1]

Представлены простые явные методы, позволяющие с малыми вычислительными затратами интегрировать жесткие системы обыкновенных дифференциальных уравнений. Рассмотрены одношаговые методы 1-го, 2-го и 3-го порядков с автоматическим выбором шага интегрирования. На тестовых задачах демонстрируется эффективность предложенных методов.

Ключевые слова: обыкновенные дифференциальные уравнения, жесткая задача Коши, явные методы, одношаговые методы, адаптивные методы.

1. Введение. Для устойчивого решения жестких систем обыкновенных дифференциальных уравнений (ОДУ) классическими явными методами необходимо выбирать очень малый шаг интегрирования, что приводит к большим вычислительным затратам. В этой связи при решении жестких задач рекомендуется применять неявные методы, в которых шаг интегрирования ограничивается только допустимой погрешностью. Однако неявные методы значительно сложнее, поскольку требуют неоднократных вычислений матрицы Якоби и решения на каждом шаге интегрирования нелинейной системы алгебраических уравнений. Сложность реализации неявных методов заставляет исследователей искать способы построения специальных явных методов, которые позволяли бы решать жесткие задачи с увеличенным (по сравнению с классическими методами) шагом интегрирования.

Существуют два основных подхода к построению явных методов для жестких задач. Первый из них основан на расширении области устойчивости и реализован в одношаговых методах типа Рунге–Кутта (см., например, [1 – 3]). Методы с расширенными областями устойчивости наиболее эффективны при решении умеренно жестких задач с равномерно заполненным спектром якобиана. Мы будем рассматривать второй подход, основанный на оценивании наибольших по модулю собственных значений якобиана и последующей стабилизации расчетной схемы в полученных точках жесткого спектра. Этот подход, применяемый в нелинейных методах, предпочтителен для задач, у которых жесткий и нежесткий спектры достаточно четко разделены (такие задачи иногда называют отделимо жесткими [3]). Явные нелинейные методы рассматривались в [4 – 6] и многих других работах, они отличаются видом нелинейности (экспоненциальные либо рациональные методы) и способом реализации (покомпонентная либо усредненная). Отметим, что в таких методах процедура оценивания собственных значений бывает “замаскирована”.

К настоящему времени накоплено большое число публикаций по явным нелинейным методам, однако предложенные ранее методы не получили широкого распространения, поскольку при решении жестких задач они существенно уступали неявным методам по точности и вычислительным затратам. Эффективный явный нелинейный метод и его программная реализация SNEIM (Single-step Nonlinear Explicit Integration Method) были предложены в [7]. Тестирование программы SNEIM показало, что она вполне может конкурировать с неявными решателями при интегрировании жестких систем с умеренной точностью (например, с помощью этой программы были успешно и с малыми вычислительными затратами решены жесткие задачи из тестового набора STIFF DETEST [8]). В [9 – 11] были изложены принципы построения одношаговых и многошаговых методов такого типа. В этих методах осуществляется самонастройка коэффициентов на решаемую задачу с использованием полученных на основе предварительных стадий оценок наибольших по модулю собственных значений матрицы Якоби (в этом смысле они были названы адаптивными).

Явные адаптивные методы были реализованы в программном комплексе “МВТУ” [12] и показали высокую эффективность, причем при решении многих жестких задач они не уступали лучшим неявным методам. Среди таких задач были пять из 12 тестов, приведенных в [3] (VDPOL, ROBER, OREGO, HIRES и CUSP). Для некоторых жестких задач явные адаптивные методы оказались даже более эффективными, чем неявные методы (примеры таких задач приведены в [9 – 12]). По итогам тестирования наиболее хорошо проявил себя явный многошаговый метод переменного порядка (от 1-го до 6-го) [11]. Этот метод довольно

[1] Московский государственный технический университет им. Н. Э. Баумана (МГТУ), факультет специального машиностроения, 2-ая Бауманская ул., д. 5, 107005, Москва; e-mail: lm_skvo@rambler.ru

сложен, поэтому в [11] приведены основные расчетные формулы, но опущены детали реализации. В то же время одношаговые адаптивные методы достаточно просты, а при умеренных требованиях к точности они показывают результаты, ненамного уступающие по эффективности многошаговому методу.

Цель настоящей статьи — показать, что существуют простые явные методы, способные эффективно, т.е. с заданной точностью и малыми вычислительными затратами, интегрировать жесткие системы ОДУ. Рассмотрены три метода (1-го, 2-го и 3-го порядков), а также приведены несложные модификации классических методов Рунге–Кутта и Мерсона, повышающие их эффективность при решении жестких задач. Приводятся все расчетные формулы, необходимые для реализации этих методов с автоматическим выбором шага, причем ключевые соотношения представлены в виде фрагментов программ на языке Паскаль.

Основная идея рассмотренных далее методов заключается в следующем. Будем интегрировать линейную систему

$$y' = Ay, \quad y_0 = y(t_0) \tag{1}$$

явным одношаговым методом типа Рунге–Кутта. Численное решение на одном шаге задается формулой $y_1 = R(hA)y_0$, где h — величина шага интегрирования и $R(z)$ — функция (многочлен) устойчивости. Пусть λ_1, λ_2, …, λ_n — собственные значения матрицы A, причем λ_1 — отрицательное число, намного превышающее по модулю остальные собственные значения. Если задать многочлен устойчивости так, чтобы выполнялось неравенство $\left|R(h\lambda_1)\right| \ll 1$, то интегрирование может быть выполнено с шагом, намного превышающем наименьшую постоянную времени $\tau = \dfrac{1}{\left|\lambda_1\right|}$. Для реализации такого подхода необходимо иметь оценку наибольшего по модулю собственного значения матрицы Якоби. Такую оценку можно получить на основе степенного метода, используя только информацию, полученную на предварительных стадиях одношагового метода.

Мы будем рассматривать методы с покомпонентной реализацией, что соответствует матричной функции устойчивости вида

$$R(hA) = I + hA + \ldots + \frac{(hA)^p}{p!} + D(hA)^{p+1}, \tag{2}$$

где p — порядок метода и D — диагональная матрица настраиваемых параметров. В этом случае по разным компонентам вектора решения могут быть получены оценки разных собственных значений, тогда функция устойчивости (2) будет стабилизирована сразу в нескольких точках жесткого спектра.

2. Метод 1-го порядка. Будем решать задачу Коши для системы ОДУ

$$y' = f(t, y), \quad y(t_0) = y_0, \tag{3}$$

где t — независимая переменная и y — n-мерный вектор. Для получения решения в следующей точке $t_1 = t_0 + h$ воспользуемся трехстадийным одношаговым методом, стадии которого выполняются согласно формулам

$$\begin{aligned} &k_0 = f(t_0, y_0), \\ &u_1 = y_0 + hk_0, \quad k_1 = f(t_1, u_1), \\ &u_2 = u_1 + h\alpha(k_1 - k_0), \quad k_2 = f(t_1, u_2), \end{aligned} \tag{4}$$

где α — достаточно малая константа (можно задать $\alpha = 10^{-3}$).

Для линейной системы (1) имеем

$$k_1 - k_0 = (hA)^2 h^{-1} y_0, \quad k_2 - k_1 = \alpha(hA)^3 h^{-1} y_0, \tag{5}$$

а в случае нелинейной системы (3) при гладкой функции f и достаточно малом h справедливы приближенные равенства

$$k_1 - k_0 \approx hAk_0, \quad k_2 - k_1 \approx \alpha(hA)^2 k_0, \tag{6}$$

где $A = \partial f/\partial y$ — матрица Якоби, вычисленная при $t = t_0$, $y = y_0$.

Соотношения (5), (6) позволяют воспользоваться степенным методом для получения оценок собственных значений матрицы hA. Обозначим i-ю компоненту вектора k через $k^{(i)}$, тогда оценку z_1 наибольшего по модулю собственного значения матрицы hA по i-й компоненте получим в виде

$$z_1 = \frac{b}{a}, \quad a = \alpha\left(k_1^{(i)} - k_0^{(i)}\right), \quad b = k_2^{(i)} - k_1^{(i)}. \tag{7}$$

Заключительную формулу для шага интегрирования по i-й компоненте принимаем в виде

$$y_1^{(i)} = u_1^{(i)} + hc\big(k_1^{(i)} - k_0^{(i)}\big), \tag{8}$$

где c — настраиваемый параметр. Зависимость c от z_1 зададим исходя из условий обеспечения наиболее высокой точности нежестких компонент и быстрого затухания жестких компонент численного решения. Для нежестких компонент (т.е. при $|z_1| \leqslant 1.6$) принимаем $c = \dfrac{1}{2} + \dfrac{z_1}{6}$. В этом случае из (7), (8) получим $y_1^{(i)} = y_0^{(i)} + h\left(k_0^{(i)} + \dfrac{1}{2}\left(k_1^{(i)} - k_0^{(i)}\right) + \dfrac{1}{6\alpha}\left(k_2^{(i)} - k_1^{(i)}\right)\right)$. Такая формула обеспечивает 2-й порядок для нелинейных систем и 3-й порядок для автономных линейных систем вида (1). Для жестких компонент (при $z_1 < -1.6$) принимаем $c = -z_1^{-1} - z_1^{-2}$, что обеспечивает равенство нулю функции устойчивости при $z = z_1$. Для неустойчивых компонент (при $z_1 > 1.6$) принимаем $c = 1.23\, z_1^{-1}$. Приведенные формулы и заданные в них константы обеспечивают правильный характер (затухание или расхождение) численного решения и непрерывную зависимость c от z_1 и от z_1^{-1}.

В зависимости от соотношения a и b в (7) будем вычислять $z_1 = \dfrac{b}{a}$ либо $z_1^{-1} = \dfrac{a}{b}$, что позволяет избежать переполнения или деления на нуль. Окончательный алгоритм реализации формулы (8) удобно записать в виде следующего фрагмента программы на языке Паскаль:

```
for i := 1 to n do
begin
  a := alpha * (k1[i] - k0[i]); b := k2[i] - k1[i];
  if abs(b) <= 1.6 * abs(a) then
  begin
    if b <> 0 then b := b/a;
    c := 1/2 + b/6;
  end else
  begin
    a := a/b;
    if a < 0 then c := -a * (1 + a) else c := 1.23 * a;
  end;
    y1[i] := u1[i] + h * c * (k1[i] - k0[i]);
end;
```

Нам осталось описать алгоритм изменения величины шага интегрирования. Будем использовать стандартную процедуру [3], согласно которой величина следующего шага принимается в виде

$$h_{\text{new}} = wh, \quad w = \text{fac}\ \text{err}^{-\gamma}, \tag{9}$$

где $\text{fac} = 0.7$ — множитель безопасности и $\gamma = 0.5$ — величина, обратная порядку оценки ошибки. Нормированная оценка ошибки err вычисляется по формуле

$$\text{err} = \max_i \frac{\left|\delta y^{(i)}\right|}{\text{Atol} + \text{Rtol} \max\left(\left|y_0^{(i)}\right|, \left|y_1^{(i)}\right|\right)}, \quad \delta y = y_1 - u_1, \tag{10}$$

где Atol — допустимая абсолютная ошибка и Rtol — допустимая относительная ошибка. При $\text{err} > 1$ шаг считается неудачным и отбрасывается, после чего производится перерасчет с уменьшенным размером шага. Рекомендуется ограничивать сверху и снизу величину w в (9) значениями $w_{\max} = 4$ и $w_{\min} = \dfrac{1}{4}$.

Формулы (4), (9), (10) и приведенный фрагмент программы полностью задают метод интегрирования с переменным шагом и могут быть использованы для его реализации в виде компьютерной программы. Метод имеет 1-й порядок для жестких задач, а для нежестких задач порядок повышается до 2-го или даже до 3-го (если задача линейная и автономная).

3. Метод 2-го порядка. Аналогично строится четырехстадийный метод 2-го порядка. Его стадии

выполняются по формулам

$$
\begin{aligned}
&k_0 = f(t_0, y_0),\\
&u_1 = y_0 + hk_0, \quad t_1 = t_0 + h, \quad k_1 = f(t_1, u_1),\\
&u_2 = u_1 + \frac{h}{2}(k_1 - k_0), \quad k_2 = f(t_1, u_2),\\
&u_3 = u_2 + h\alpha(k_2 - k_1), \quad k_3 = f(t_1, u_3).
\end{aligned}
$$

Первые три стадии задают метод Хойна, на основе которого и строится наш метод, а четвертая стадия необходима для получения оценок наибольшего собственного значения. Шаг интегрирования выполняется согласно следующему алгоритму:

```
for i := 1 to n do
begin
  a := alpha * (k2[i] - k1[i]); b := k3[i] - k2[i];
  if abs(b) <= 2 * abs(a) then
  begin
    if b <> 0 then b := b/a;
    c := 1/3 + b/12;
  end else
  begin
    a := a/b;
    if a < 0 then c := a * (1 + a)/(a - 1) else c := a;
  end;
  y1[i] := u2[i] + h * c * (k2[i] - k1[i]);
end;
```

Оценивание ошибки и изменение величины шага выполняются по формулам (9), (10). Метод имеет 2-й порядок точности для жестких задач, а для автономных линейных нежестких задач порядок повышается до 4-го.

4. Метод 3-го порядка. При попытках построения адаптивных методов более высоких порядков мы столкнулись с тем, что реальный порядок точности при решении жестких задач оказывался ниже, чем можно было бы ожидать исходя из классических представлений. Применительно к неявным методам Рунге–Кутта такое явление получило известность как “феномен снижения порядка” [3]. Оказалось, что аналогичное явление возникает также и при решении жестких задач явными методами. В [13, 14] были предложены приемы, позволяющие избежать снижения точности при решении жестких задач стабилизированными явными методами (в том числе и адаптивными). Эти приемы сводятся к различным способам минимизации норм некоторых функций, которые были названы функциями погрешности.

Для построения метода, реально обеспечивающего 3-й порядок точности при решении жестких задач, необходимо минимизировать норму функции погрешности $e_2(z)$ [13]. Мы взяли за основу четырехстадийный метод 3-го порядка, имеющий $e_2(z) \equiv 0$ [14]. Первые пять стадий выполняются согласно стадиям и заключительной формуле этого метода, а шестая стадия необходима для получения оценок собственных значений. В результате получаем шестистадийный метод, стадии которого выполняются по формулам

$$
\begin{aligned}
&k_0 = f(t_0, y_0),\\
&u_1 = y_0 + \frac{h}{2}k_0, \quad k_1 = f\left(t_0 + \frac{h}{2}, u_1\right),\\
&u_2 = y_0 + hk_0, \quad t_1 = t_0 + h, \quad k_2 = f(t_1, u_2),\\
&u_3 = y_0 + h\left(2k_1 - \frac{k_0 + k_2}{2}\right), \quad k_3 = f(t_1, u_3),\\
&u_4 = y_0 + \frac{h}{6}(k_0 + 4k_1 - k_2 + 2k_3), \quad k_4 = f(t_1, u_4),\\
&u_5 = u_4 + h\alpha(k_4 - k_3), \quad k_5 = f(t_1, u_5).
\end{aligned}
$$

Шаг интегрирования выполняется согласно следующему алгоритму:

```
for i := 1 to n do
begin
  a := alpha * (k4[i] - k3[i]); b := k5[i] - k4[i];
  if abs(b) <= 2.2 * abs(a) then
  begin
    if b <> 0 then b := b/a;
    c := 1/4 + b/20;
  end else
  begin
    a := a/b;
    if a < 0 then c := -a * (a * (a * (6 * a + 6) + 3) + 1) else c := 0.792 * a;
  end;
  y1[i] := u4[i] + h * c * (k4[i] - k3[i]);
end;
```

Как и в предыдущих методах, управление шагом осуществляется по формулам (9), (10), но в (9) принимаем $\gamma = \dfrac{1}{3}$, а в (10) используем оценку ошибки в виде $\delta y = y_1 - u_3$. Метод имеет 3-й порядок для жестких задач, а для нежестких задач порядок может повышаться до 4-го или даже до 5-го.

5. Модификации классических методов. Принцип построения адаптивных методов может быть использован также и для улучшения хорошо известных явных методов Рунге–Кутта. Покажем это на примерах классического метода Рунге–Кутта и метода Мерсона. Эти методы реализованы во многих библиотеках и пакетах программ (например, в Библиотеке численного анализа НИВЦ МГУ [15] и в программном комплексе "МВТУ" [12]). Для усовершенствования программных реализаций этих методов достаточно добавить несколько дополнительных операторов, в результате чего заметно повышается эффективность решения жестких задач. На результатах решения нежестких задач эти изменения практически не сказываются. Таким образом, можно заметно расширить область эффективного применения явных решателей ОДУ.

Классический метод Рунге–Кутта 4-го порядка задается формулами

$$
\begin{aligned}
&k_0 = f(t_0, y_0),\\
&u_1 = y_0 + \frac{h}{2}\,k_0, \quad k_1 = f\left(t_0 + \frac{h}{2}\,, u_1\right),\\
&u_2 = y_0 + \frac{h}{2}\,k_1, \quad k_2 = f\left(t_0 + \frac{h}{2}\,, u_2\right),\\
&u_3 = y_0 + hk_2, \quad k_3 = f(t_0 + h, u_3),\\
&y_1 = y_0 + \frac{h}{6}\,(k_0 + 2k_1 + 2k_2 + k_3).
\end{aligned}
$$

Все изменения относятся только к формуле вычисления u_3, которая в модифицированном методе задается в виде следующей последовательности операторов:

```
for i := 1 to n do
begin
  a := k1[i] - k0[i]; b := k2[i] - k1[i];
  if (abs(b) <= abs(a)) or (a * b >= 0) then u3[i] := y0[i] + h * k2[i]
  else
  begin
    b := a/b; b := -(1 + 3 * b * (1 + b + 0.5 * b * b));
    u3[i] := y0[i] + h * (k0[i] + b * a);
  end;
end;
```

Метод Мерсона 4-го порядка задается формулами

$$k_0 = f(t_0, y_0),$$
$$u_1 = y_0 + \frac{h}{3}\,k_0, \quad k_1 = f\left(t_0 + \frac{h}{3}\,, u_1\right),$$
$$u_2 = y_0 + \frac{h}{6}\,(k_0 + k_1), \quad k_2 = f\left(t_0 + \frac{h}{3}\,, u_2\right),$$
$$u_3 = y_0 + \frac{h}{8}\,(k_0 + 3k_2), \quad k_3 = f\left(t_0 + \frac{h}{2}\,, u_3\right),$$
$$u_4 = y_0 + \frac{h}{2}\,(k_0 - 3k_2 + 4k_3), \quad k_4 = f(t_0 + h, u_4),$$
$$y_1 = y_0 + \frac{h}{6}\,(k_0 + 4k_3 + k_4).$$

Модифицированный метод Мерсона также отличается формулой вычисления u_3, которая задается операторами

```
for i := 1 to n do
begin
  a := k1[i] - k0[i]; b := k2[i] - k1[i];
  if (2 * abs(b) <= abs(a)) or (a * b >= 0) then
    u3[i] := y0[i] + (h/8) * (k0[i] + 3 * k2[i])
  else
  begin
    b := a/(6 * b); b := 0.125 - b * (1 + b * (2.5 + b * (4 + b)));
    u3[i] := y0[i] + h * (0.5 * k0[i] + b * a);
  end;
end;
```

6. Результаты решения тестовых задач. Приведем результаты решения пяти жестких задач, рассмотренных в [3]. В качестве показателя точности использовалось значение

$$\mathrm{scd} = -\lg\left(\max_i\left(\frac{|y_i - \widetilde{y}_i|}{|y_i|}\right)\right),$$

где y_i — точное, а $\widetilde{y}_i$ — численное решение по i-й компоненте в конечной точке интервала интегрирования. Таким образом, scd (significant correct digits) — число правильных значащих цифр численного решения.

Вычислительные затраты оценивались числом вычислений правой части Nf. Обозначим через A1, A2 и A3 рассмотренные методы 1-го, 2-го и 3-го порядков. Для сравнения приведем результаты наиболее эффективного явного адаптивного метода [11], который обозначим через A4 (в программном комплексе "МВТУ" этот метод назван "Адаптивный 4"), а также метода Мерсона (классического и модифицированного).

Задача VDPOL (осциллятор Ван-дер-Поля):

$$y_1' = y_2, \quad y_2' = 10^6\big((1 - y_1^2)y_2 - y_1\big),$$
$$y_1(0) = 2, \quad y_2(0) = 0, \quad 0 \leqslant t \leqslant 2.$$

Результаты решения этой задачи при $h_0 = 10^{-6}$ и Atol = Rtol приведены в табл. 1.

Задача OREGO (орегонатор; модель, описывающая реакцию Белоусова–Жаботинского):

$$y_1' = 77.27\big(y_2 + y_1(1 - 8.375 \times 10^{-6}y_1 - y_2)\big), \quad y_2' = \frac{1}{77.27}\left(y_3 - (1 + y_1)y_2\right), \quad y_3' = 0.161(y_1 - y_3),$$
$$y_1(0) = 1, \quad y_2(0) = 2, \quad y_3(0) = 3, \quad 0 \leqslant t \leqslant 360.$$

Результаты решения этой задачи при $h_0 = 10^{-2}$ и Atol = Rtol приведены в табл. 2.

Таблица 1

Результаты решения задачи VDPOL

Метод	$\mathrm{Rtol} = 10^{-2}$		$\mathrm{Rtol} = 10^{-3}$		$\mathrm{Rtol} = 10^{-4}$	
	scd	Nf	scd	Nf	scd	Nf
A1	1.37	2 338	1.94	7 744	2.63	25 870
A2	2.96	9 675	4.23	15 403	5.16	34 651
A3	3.59	24 638	4.87	27 411	5.63	30 128
A4	3.14	710	3.23	964	4.11	1 384
Мерсона модиф.	2.70	356 236	4.34	382 909	5.43	399 550
Мерсона	2.87	5 407 932	4.42	5 409 612	5.01	5 410 724

Таблица 2

Результаты решения задачи OREGO

Метод	$\mathrm{Rtol} = 10^{-2}$		$\mathrm{Rtol} = 10^{-3}$		$\mathrm{Rtol} = 10^{-4}$	
	scd	Nf	scd	Nf	scd	Nf
A1	0.12	2 746	0.46	8 100	1.16	25 470
A2	1.50	8 929	2.38	11 908	3.42	32 437
A3	2.39	21 623	3.16	23 325	3.84	27 149
A4	1.72	1 999	2.50	2 374	3.57	2 912
Мерсона модиф.	2.62	770 888	4.65	784 437	6.18	834 313
Мерсона	3.41	15 877 370	4.72	15 877 908	5.94	15 879 053

Задача HIRES (химическая реакция с участием восьми реагентов):

$$
\begin{aligned}
&y_1' = -1.71y_1 + 0.43y_2 + 8.32y_3 + 0.0007, \quad && y_2' = 1.71y_1 - 8.75y_2,\\
&y_3' = -10.03y_3 + 0.43y_4 + 0.035y_5, && y_4' = 8.32y_2 + 1.71y_3 - 1.12y_4,\\
&y_5' = -1.745y_5 + 0.43y_6 + 0.43y_7, && y_6' = -280y_6y_8 + 0.69y_4 + 1.71y_5 - 0.43y_6 + 0.69y_7,\\
&y_7' = 280y_6y_8 - 1.81y_7, && y_8' = -y_7',\\
&y_1(0) = 1, \quad y_2(0) = \ldots = y_7(0) = 0, \quad y_8(0) = 0.0057, \quad 0 \leqslant t \leqslant 321.8122.
\end{aligned}
$$

Результаты решения этой задачи при $h_0 = 10^{-2}$ и $\mathrm{Atol} = 10^{-4}\,\mathrm{Rtol}$ приведены в табл. 3.

Таблица 3

Результаты решения задачи HIRES

Метод	$\mathrm{Rtol} = 10^{-2}$		$\mathrm{Rtol} = 10^{-3}$		$\mathrm{Rtol} = 10^{-4}$	
	scd	Nf	scd	Nf	scd	Nf
A1	0.86	1 116	2.47	2 559	2.79	7 247
A2	1.87	1 951	2.51	3 742	4.19	9 927
A3	2.76	2 639	3.70	2 859	4.22	3 770
A4	1.55	1 151	3.21	1 367	3.54	1 674
Мерсона модиф.	3.55	8 096	3.98	9 595	4.81	14 433
Мерсона	2.89	48 544	5.48	48 652	6.39	49 222

Задача CUSP (комбинация модели прохождения нервного импульса и осциллятора Ван-дер-Поля):

$$
\begin{aligned}
&y_i' = -10^4\big(y_i^3 + a_i y_i + b_i\big) + D(y_{i-1} - 2y_i + y_{i+1}),\\
&a_i' = b_i + 0.07 v_i + D(a_{i-1} - 2a_i + a_{i+1}),\\
&b_i' = \big(1 - a_i^2\big) b_i - a_i - 0.4 y_i + 0.035 v_i + D(b_{i-1} - 2b_i + b_{i+1}),\\
&v_i = \frac{u_i}{u_i + 0.1}, \quad u_i = (y_i - 0.7)(y_i - 1.3), \quad D = \frac{N^2}{144},\\
&y_0 = y_N, \quad a_0 = a_N, \quad b_0 = b_N, \quad y_{N+1} = y_1, \quad a_{N+1} = a_1, \quad b_{N+1} = b_1,\\
&y_i(0) = 0, \quad a_i(0) = -2\cos\left(\frac{2i\pi}{N}\right), \quad b_i(0) = 2\sin\left(\frac{2i\pi}{N}\right), \quad i = 1, \ldots, N, \quad N = 32, \quad 0 \leqslant t \leqslant 1.1.
\end{aligned}
$$

Результаты решения этой задачи при $h_0 = 10^{-5}$ и $\mathrm{Atol} = 10^{-2}\,\mathrm{Rtol}$ приведены в табл. 4.

Задача BRUSS (получена в результате дискретизации уравнений в частных производных, описывающих химическую реакцию с диффузией):

$$
\begin{aligned}
&u_i' = 1 + u_i^2 v_i - 4u_i + \frac{\alpha}{\Delta x^2}(u_{i-1} - 2u_i + u_{i+1}),\\
&v_i' = 3u_i - u_i^2 v_i + \frac{\alpha}{\Delta x^2}(v_{i-1} - 2v_i + v_{i+1}),\\
&u_0 = u_{N+1} = 1, \quad v_0 = v_{N+1} = 3, \quad \alpha = \frac{1}{50}, \quad \Delta x = \frac{1}{N+1},\\
&u_i(0) = 1 + \sin(2\pi x_i), \quad v_i(0) = 3, \quad x_i = i\Delta x, \quad i = 1, \ldots, N, \quad N = 100, \quad 0 \leqslant t \leqslant 10.
\end{aligned}
$$

Результаты решения этой задачи при $h_0 = 10^{-3}$ и $\mathrm{Atol} = \mathrm{Rtol}$ приведены в табл. 5.

Таблица 4

Результаты решения задачи CUSP

Метод	$\mathrm{Rtol} = 10^{-2}$		$\mathrm{Rtol} = 10^{-3}$		$\mathrm{Rtol} = 10^{-4}$	
	scd	Nf	scd	Nf	scd	Nf
A1	2.10	1 855	2.40	4 832	3.60	12 898
A2	4.44	14 350	4.09	8 138	4.87	12 899
A3	4.08	7 667	3.49	7 576	5.53	8 700
A4	2.75	1 576	3.12	1 222	4.45	1 186
Мерсона модиф.	5.54	20 355	6.01	14 234	7.42	20 512
Мерсона	3.86	96 869	4.34	97 014	6.31	97 312

Таблица 5

Результаты решения задачи BRUSS

Метод	$\mathrm{Rtol} = 10^{-2}$		$\mathrm{Rtol} = 10^{-3}$		$\mathrm{Rtol} = 10^{-4}$	
	scd	Nf	scd	Nf	scd	Nf
A1	1.00	2 396	1.88	2 621	2.26	3 386
A2	2.84	3 993	3.73	4 037	4.42	4 493
A3	3.07	6 307	4.19	6 503	4.94	6 442
A4	2.63	4 063	3.50	3 936	4.12	3 683
Мерсона модиф.	3.86	4 970	5.35	5 138	5.72	6 943
Мерсона	4.58	11 494	4.65	11 494	4.90	11 499

Обсудим полученные результаты. По сравнению с методом Мерсона явные адаптивные методы требуют существенно меньших вычислительных затрат, давая при этом качественно правильные результаты. Относительно малые значения scd для задач OREGO и HIRES объясняются тем, что в окрестности конечной точки интервала интегрирования некоторые компоненты решения очень быстро изменяются. Задачи

VDPOL и OREGO очень жесткие, и именно для них адаптивные методы обеспечивают наибольший выигрыш. Задачи CUSP и BRUSS отличаются высокой размерностью (96 и 200 уравнений соответственно) и большим числом жестких собственных значений (т.е. собственных значений с большими по модулю отрицательными вещественными частями). Задача CUSP имеет 32 жестких собственных значения, и все же адаптивные методы оказались для нее весьма эффективными. Задача BRUSS имеет равномерно заполненный спектр якобиана, что объясняет сравнительно небольшое преимущество адаптивных методов при ее решении. Отметим превосходные результаты адаптивного многошагового метода переменного порядка A4 [11], который при решении многих жестких задач не уступает лучшим неявным методам.

СПИСОК ЛИТЕРАТУРЫ

1. *Лебедев В.И.* Как решать явными методами жесткие системы дифференциальных уравнений // Вычислительные процессы и системы. Вып. 8. М.: Наука, 1991. 237–291.
2. *Новиков Е.А.* Явные методы для жестких систем. Новосибирск: Наука, 1997.
3. *Хайрер Э., Ваннер Г.* Решение обыкновенных дифференциальных уравнений. Жесткие и дифференциально-алгебраические задачи. М.: Мир, 1999.
4. *Fowler M.E., Warten R.M.* A numerical integration technique for ordinary differential equations with widely separated eigenvalues // IBM J. Research and Development. 1967. **11**, N 5. 537–543.
5. *Бобков В.В.* Новые явные A-устойчивые методы численного решения дифференциальных уравнений // Дифференциальные уравнения. 1978. **14**, № 12. 2249–2251.
6. *Заворин А.Н.* Применение нелинейных методов для расчета переходных процессов в электрических цепях // Изв. вузов. Радиоэлектроника. 1983. **26**, № 3. 35–41.
7. *Скворцов Л.М.* Расчет на ЭВМ линейных САУ. М.: Изд-во МВТУ, 1987.
8. *Enright W.H., Hull T.E., Lindberg B.* Comparing numerical methods for stiff systems of ODEs // BIT. 1975. **15**, N 1. 10–48.
9. *Скворцов Л.М.* Адаптивные методы численного интегрирования в задачах моделирования динамических систем // Изв. РАН. Теория и системы управления. 1999. № 4. 72–78.
10. *Скворцов Л.М.* Явные адаптивные методы численного решения жестких систем // Матем. моделирование. 2000. **12**, № 12. 97–107.
11. *Скворцов Л.М.* Явный многошаговый метод численного решения жестких дифференциальных уравнений // Журн. вычисл. матем. и матем. физики. 2007. **47**, № 6. 959–967.
12. *Козлов О.С., Скворцов Л.М., Ходаковский В.В.* Решение дифференциальных и дифференциально-алгебраических уравнений в программном комплексе “МВТУ” (http://model.exponenta.ru/mvtu/20051121.html).
13. *Скворцов Л.М.* Точность методов Рунге–Кутты при решении жестких задач // Журн. вычисл. матем. и матем. физики. 2003. **43**, № 9. 1374–1384.
14. *Скворцов Л.М.* Явные методы Рунге–Кутты для умеренно жестких задач // Журн. вычисл. матем. и матем. физики. 2005. **45**, № 11. 2017–2030.
15. *Арушанян О.Б., Залеткин С.Ф.* Общее описание подпрограмм решения обыкновенных дифференциальных уравнений Библиотеки численного анализа НИВЦ МГУ // Вычислительные методы и программирование. 2003. **4**, № 1. 213–221 (http://num-meth.srcc.msu.ru).

Поступила в редакцию
27.03.2008