9. Решение обыкновенных дифференциальных уравнений и систем

Дифференциальные уравнения и системы описывают очень многие динамические процессы и возникают при решении различных задач физики, электротехники, химии и других наук. Данная глава посвящена численному решению дифференциальных уравнений и систем средствами Octave.

9.1 Общие сведения о дифференциальных уравнениях

Дифференциальным уравнением n-го порядка называется соотношение вида

$$H\left(t, x, x', x'', \ldots, x^{(n)}\right)=0. \tag{9.1}$$

Решением дифференциального уравнения называется функция $x(t)$, которая обращает уравнение в тождество.

Системой дифференциальных уравнений n-го порядка называется система вида:

$$\begin{cases} x_1' = f_1(t, x_1, x_2, \ldots, x_n) \\ x_2' = f_2(t, x_1, x_2, \ldots, x_n) \\ \ldots\ldots\ldots\ldots \\ x_n' = f_n(t, x_1, x_2, \ldots, x_n) \end{cases} \tag{9.2}$$

Системой линейных дифференциальных уравнений называется система вида:

$$\begin{cases} x_1' = \sum_{j=1}^{n} a_{1j} x_j + b_1 \\ x_2' = \sum_{j=1}^{n} a_{2j} x_j + b_2 \\ \ldots\ldots\ldots\ldots \\ x_n' = \sum_{j=1}^{n} a_{nj} x_j + b_n \end{cases} \tag{9.3}$$

Решением системы называется вектор $x(t)=\begin{cases} x_1(t) \\ x_2(t) \\ \ldots \\ x_n(t) \end{cases}$, который обращает уравнения систем (9.2), (9.3) в тождества.

Каждое дифференциальное уравнение, так же, как и система, имеет бесконечное множество решений, которые отличаются друг от друга константами. Для однозначного определения решения необходимо определить дополнительные начальные или граничные условия. Количество таких условий должно совпадать с порядком дифференциального уравнения или системы. В зависимости от вида дополнительных условий в дифференциальных уравнениях различают:

- задачу Коши, в случае, если все дополнительные условия заданы в одной (чаще начальной) точке интервала;

- краевую задачу, в случае, когда дополнительные условия заданы на границах интервала.

Различают *точные* (аналитические) и *приближенные* (численные) методы решения дифференциальных уравнений. Большое количество уравнений может быть решено точно. Однако есть уравнения, а особенно системы уравнений, для которых нельзя записать точное решение. Но даже для уравнений с известным аналитическим решением очень часто необходимо вычислить числовое значение при определенных исходных данных. Поэтому широкое распространение получили численные методы решения обыкновенных дифференциальных уравнений.

9.2 Численные методы решения дифференциальных уравнений и их реализация в *Octave*

Численные методы решения дифференциального уравнения первого порядка будем рассматривать для следующей задачи Коши. Найти решение дифференциального уравнения

$$x' = f(x, t) \tag{9.4}$$

удовлетворяющее начальному условию

$$x(t_0) = x_0 \tag{9.5}$$

иными словами, требуется найти интегральную кривую $x = x(t)$, проходящую через заданную точку $M_0(t_{0,} x_0)$ (рис. 9.1).

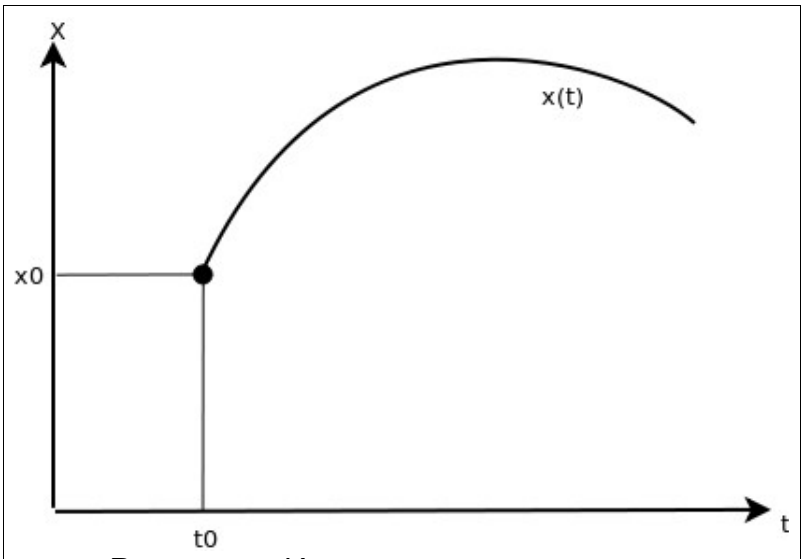

Рисунок *9.1*. Интегральная кривая , проходящая через точку $M_0(t_{0,} x_0)$

Для дифференциального уравнения n-го порядка

$$x^{(n)} = f(t, x, x', x'', \ldots, x^{(n-1)}) \tag{9.6}$$

задача Коши состоит в нахождении решения $x = x(t)$, удовлетворяющего уравнению (9.6) и начальным условиям

$$x(t_0)=x_0, x'(t_0)=x'_0, \dots, x^{(n-1)}(t_0)=x_0^{(n-1)}. \quad (9.7)$$

Рассмотрим основные численные методы решения задачи Коши.

9.2.1 Решение дифференциальных уравнений методом Эйлера

При решении задачи Коши (9.4), (9.5) на интервале $[t_0, t_n]$, выбрав достаточно малый шаг *h*, построим систему равноотстоящих точек

$$h=\frac{t_n-t_0}{n}, t_i=t_0+ih, i=0,n \quad (9.8)$$

Для вычисления значения функции в точке t_1 разложим функцию $x=x(t)$ в окрестности точки t_0 в ряд Тейлора [2]

$$x(t_1)=x(t_0+h)=x(t_0)+x'(t_0)h+x''(t_0)\frac{h^2}{2}+\dots \quad (9.9)$$

При достаточном малом значении *h* членами выше второго порядка можно пренебречь и с учетом $x'(t_0)=f(x_0, t_0)$ получим следующую формулу для вычисления приближенного значения функции $x(t)$ в точке t_1

$$h=\frac{t_n-t_0}{n}, t_i=t_0+ih, i=0,n \quad (9.10)$$

Рассматривая найденную точку (x_1, t_1), как начальное условие задачи Коши запишем аналогичную формулу для нахождения значения функции $x(t)$ в точке t_2

$$x_2=x_1+h f(x_1, t_1).$$

Повторяя этот процесс, сформируем последовательность значений x_i в точках t_i по формуле

$$x_{i+1}=x_i+h f(x_i, t_i), i=1,n. \quad (9.11)$$

Процесс нахождения значений функции x_i в узловых точках t_i по формуле (9.11) называется *методом Эйлера*. Геометрическая интерпретация метода Эйлера состоит в замене интегральной кривой $x(t)$ ломаной $M_0 M_1 M_2 \dots, M_n$ с вершинами $M_i(x_i, y_i)$. Звенья ломанной Эйлера $M_i M_{i+1}$ в каждой вершине M_i имеют направление $y_i=f(t_i, x_i)$, совпадающее с направлением интегральной кривой $x(t)$ уравнения (9.4), проходящей через точку M_i (рис. 9.2). Последовательность ломанных Эйлера при $h \to 0$ на достаточно малом отрезке $[x_i, x_i+h]$ стремится к искомой интегральной кривой.

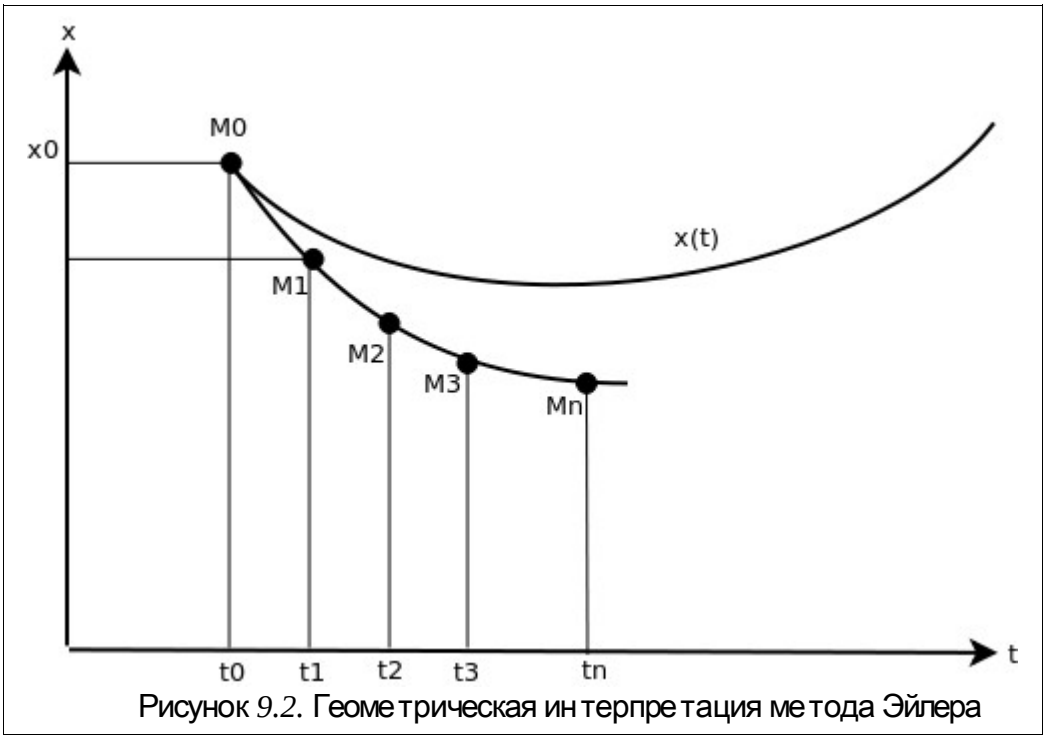

Рисунок *9.2*. Геометрическая интерпретация метода Эйлера

На каждом шаге решение $x(t)$ определяется с ошибкой за счет отбрасывания членов ряда Тейлора выше первой степени, что в случае быстроменяющейся функции $f(t,x)$ может привести к быстрому накапливанию ошибки. В методе Эйлера следует выбирать достаточной малый шаг *h*.

9.2.2 Решение дифференциальных уравнений при помощи модифицированного метода Эйлера

Более точным методом решения задачи (9.4)-(9.5) является *модифицированный метод Эйлера*, при котором сначала вычисляют промежуточные значения [2]

$$t_p = t_i + \frac{h}{2}, x_p = x_i + \frac{h}{2} f(x_i, t_i) \tag{9.12}$$

после чего находят значение x_{i+1} по формуле

$$x_{i+1} = x i + h f(x_p, t_p), i = 1, n \tag{9.13}$$

9.2.3 Решение дифференциальных уравнений методами Рунге-Кутта

Рассмотренные выше методы Эйлера (как обычный, так и модифицированный) являются частными случаями явного *метода Рунге-Кутта* k-го порядка. В общем случае формула вычисления очередного приближения методом Рунге-Кутта имеет вид [2]:

$$x_{i+1} = x_i + h\phi(t_i, x_i, h), i = 1, n \tag{9.14}$$

Функция $\phi(t,x,h)$ приближает отрезок ряда Тейлора до k-го порядка и не

содержит частных производных $f(t,x)$ [2].

Метод Эйлера является *методом Рунге-Кутта первого порядка* (k=1) и получается при $\phi(t,x,h)=f(t,x)$.

1. Семейство *методов Рунге-Кутта второго порядка* имеет вид [2]

$$x_{i+1}=x_i+((1-\alpha)f(t_i,x_i)+\alpha f(t_i+\frac{h}{2\alpha},x_i+\frac{h}{2\alpha}f(t_i,x_i))),i=1,n \qquad (9.15)$$

Два наиболее известных среди методов Рунге-Кутта второго порядка [2] – это метод Хойна ($\alpha=\frac{1}{2}$) и модифицированный метод Эйлера ($\alpha=1$).

Подставив $\alpha=\frac{1}{2}$ в формулу (9.15), получаем расчетную формулу *метода Хойна* [2]:

$$x_{i+1}=x_i+\frac{h}{2}f(t_i,x_i)+f(t_i+h,x_i+hf(t_i,x_i)),i=1,n \qquad (9.16)$$

Подставив $\alpha=1$ в формулу (9.15), получаем расчетную формулу уже рассмотренного выше модифицированного метода Эйлера

$$x_{i+1}=x_i+h(f(t_i+\frac{h}{2},x_i+\frac{h}{2}f(t_i,x_i))),i=1,n \qquad (9.17)$$

Наиболее известным является *метод Рунге-Кутта четвертого порядка*, расчетные формулы которого можно записать в виде [2]:

$$\begin{cases} x_{i+1}=x_i+\Delta x_i, i=1,n \\ \Delta x_i=\frac{h}{6}(K_1^i+2K_2^i+2K_3^i+K_4^i) \\ K_1^i=f(t_i,x_i) \\ K_2^i=f(t_i+\frac{h}{2},x_i+\frac{h}{2}K_1^i) \\ K_3^i=f(t_i+\frac{h}{2},x_i+\frac{h}{2}K_2^i) \\ K_4^i=f(t_i+h,x_i+hK_3^i) \end{cases} \qquad (9.18)$$

Одной из модификаций метода Рунге-Кутта является *метод Кутта-Мерсона* (или *пятиэтапный метод Рунге-Кутта четвертого порядка*), который состоит в следующем [2].

1. На *i*-м шаге рассчитываются коэффициенты

$$
\begin{aligned}
K_1^i &= f(t_i, x_i) \\
K_2^i &= f(t_i + \frac{h}{3}, x_i + \frac{3}{2} K_1^i) \\
K_3^i &= f(t_i + \frac{h}{3}, x_i + \frac{h}{6} K_1^i + \frac{h}{6} K_2^i) \\
K_4^i &= f(t_i + \frac{h}{2}, x_i + \frac{h}{8} K_1^i + \frac{3h}{2} K_2^i) \\
K_5^i &= f(t_i + h, x_i + \frac{h}{2} K_1^i - \frac{3h}{2} K_3^i + 2h K_4^i)
\end{aligned}
\tag{9.19}
$$

2. Вычисляем приближенное значение $x(t_{i+1})$ по формуле

$$x_{i+1}^{\sim} = x_i + \frac{h}{2}(K_1^i - 3K_3^i + 4K_4^i) \tag{9.20}$$

3. Вычисляем приближенное значение $x(t_{i+1})$ по формуле

$$x_{i+1} = x_i + \frac{h}{6}(K_1^i + 4K_3^i + K_5^i) \tag{9.21}$$

4. Вычисляем оценочный коэффициент по формуле

$$R = 0.2|x_{i+1} - x_{i+1}^{\sim}| \tag{9.22}$$

5. Сравниваем R с точностью вычислений ε. Если $R \geq \varepsilon$, то уменьшаем шаг вдвое и возвращаемся к п.1. Если $R < \varepsilon$, то значение, вычисленное по формуле (9.21), и будет вычисленным значением $x(t_{i+1})$ (с точностью ε).

6. Перед переходом к вычислению следующего значения x, сравниваем R с $\frac{\varepsilon}{64}$. Если $R < \frac{\varepsilon}{64}$, то дальнейшие вычисления можно проводить с удвоенным шагом $h = 2h$.

Рассмотренные методы Рунге-Кутта относятся к классу одношаговых методов, в которых для вычисления значения в очередной точке x_{k+1} нужно знать значение в предыдущей точке x_k.

Еще один класс методов решения задачи Коши – *многошаговые методы*, в которых используются точки x_{k-3}, x_{k-2}, x_{k-1}, x_k для вычисления x_{k+1}. В многошаговых методах первые четыре начальные точки (t_0, x_0), (t_1, x_1), (t_2, x_2), (t_3, x_3) должны быть получены заранее любым из одношаговых методов (метод Эйлера, Рунге-Кутта и т.д.). Наиболее известными *многошаговыми методами* являются методы *прогноза-коррекции Адамса* и *Милна*.

9.2.4 Решение дифференциальных уравнений методом прогноза-коррекции Адамса

Рассмотрим решение уравнения (9.1)-(9.2) на интервале $[t_i, t_{i+1}]$. Будем считать, что решение в точках $t_0, t_1, t_2, \ldots, t_i$ уже найдено, и значения в этих точках будем использовать для нахождения значения $x(t_{i+1})$.

Проинтегрируем уравнение (9.1) на интервале $[t_i, t_{i+1}]$ и получим соотношение [2]

$$x(t_{i+1})=x(t_i)+\int_{t_i}^{t_{i+1}} f(t,x(t))\,dt \tag{9.23}$$

При вычислении интеграла, входящего в (9.23), вместо функции $f(t,x(t))$ будем использовать интерполяционный полином Лагранжа, построенный по точкам (t_{i-3},x_{i-3}), (t_{i-2},x_{i-2}), (t_{i-1},x_{i-1}), (t_i,x_i). Подставив полином Лагранжа в (9.23), получаем первое приближение (прогноз) $\tilde{x}_{i+1}$ для значения функции в точке t_{i+1}

$$\tilde{x}_{i+1}=x_i+\frac{h}{24}(-9f(t_{i-3},x_{i-3})+37f(t_{i-2},x_{i-2})-59f(t_{i-1},x_{i-1})+55f(t_i,x_i)) \tag{9.24}$$

Как только $\tilde{x}_{i+1}$ вычислено, его можно использовать. Следующий полином Лагранжа для функции $f(t,x(t))$ построим по точкам (t_{i-2},x_{i-2}), (t_{i-1},x_{i-1}), (t_i,x_i) и новой точке $(t_{i+1},\tilde{x}_{i+1})$, после чего подставляем его в (9.23) и получаем второе приближение (корректор)

$$x_{i+1}=x_i+\frac{h}{24}(f(t_{i-2},x_{i-2})-5f(t_{i-1},x_{i-1})+19f(t_i,x_i)+9f(t_{i+1},\tilde{x}_{i+1})) \tag{9.25}$$

Таким образом, для вычисления значения $x(t_{i+1})$ методом Адамса необходимо последовательно применять формулы (9.24), (9.25) [2], а первые четыре точки можно получить методом Рунге-Кутта.

9.2.5 Решение дифференциальных уравнений методом Милна

Отличие *метода Милна* от метода Адамса состоит в использовании в качестве интерполяционного полинома Ньютона.

Подставив в (9.23) вместо функции $f(t,x(t))$ интерполяционный полином Ньютона, построенный по точкам (t_{k-3},x_{k-3}), (t_{k-2},x_{k-2}), (t_{k-1},x_{k-1}), (t_k,x_k) получаем первое приближение – прогноз Милна $\tilde{x}_{k+1}$ для значения функции в точке t_{k+1} [2]

$$\tilde{x}_{k+1}=x_{k-3}+\frac{4h}{3}(2f(t_{k-2},x_{k-2})-f(t_{k-1},x_{k-1})+2f(t_k,x_k)) \tag{9.26}$$

Следующий полином Ньютона для функции $f(t,x(t))$ построим по точкам (t_{k-2},x_{k-2}), (t_{k-1},x_{k-1}), (t_k,x_k) и новой точке $(t_{k+1},\tilde{x}_{k+1})$, после чего подставляем его в (9.23) и получаем второе приближение – корректор Милна [2]

$$x_{k+1}=x_{k-1}+\frac{h}{3}(f(t_{k-1},x_{k-1})+4f(t_k,x_k)+f(t_{k+1},\tilde{x}_{k+1})) \tag{9.27}$$

В методе Милна для вычисления значения $x(t_{k+1})$ необходимо последовательно применять формулы (9.26), (9.27), а первые четыре точки можно получить методом Рунге-Кутта.

Существует *модифицированный метод Милна*. В нем сначала вычисляется первое

приближение по формуле (9.26), затем вычисляется управляющий параметр [2]

$$m_{k+1}=\tilde{x_{k+1}}+\frac{28}{29}(x_k-\tilde{x_k}) \tag{9.28}$$

после чего вычисляется значение второго приближения – корректор Милна по формуле

$$x_{k+1}=x_{k-1}+\frac{h}{3}(f(t_{k-1},x_{k-1})+4f(t_k,x_k)+f(t_{k+1},m_{k+1})) \tag{9.29}$$

В модифицированном методе Милна первые четыре точки можно получить методом Рунге-Кутта, а для вычисления значения $x(t_{k+1})$ необходимо последовательно применять формулы (9.26), (9.28), (9.29).

9.3 Реализация численных методов в *Octave*

Ниже приведены тексты функций, реализующие рассмотренных в п. 9.2 , численных методов решения дифференциальных уравнений.

```
function [x,t]=eiler(a,b,n,x0)
%Функция решения задачи Коши x'(t)=g(t,x) x(a)=x0 методом
%Эйлера на интервале интегрирования [a,b], n — количество
%отрезков, на которые разбивается интервал [a,b].
%Вычисление шага h.
h=(b-a)/n;
x(1)=x0;
% Формирование системы равноотстоящих узлов ti
for i=1:n+1
t(i)=a+(i-1)*h;
end
%Вычисление значение функции в узловых точках методом
%Эйлера по формуле (9.11)
for i=2:n+1
x(i)=x(i-1)+h*g(t(i-1),x(i-1));
end
end
```

Листинг 9.1. Функция решения задачи Коши методом Эйлера.

```
function [x,t]=eiler_m(a,b,n,x0)
%Функция решения задачи Коши x'(t)=g(t,x) x(a)=x0
%модифицированным методом Эйлера на интервале
% интегрирования [a,b], n - количество отрезков, на
% которые разбивается интервал [a,b].
% Вычисление шага h.
h=(b-a)/n;
x(1)=x0;
% Формирование системы равноотстоящих узлов ti
for i=1:n+1
t(i)=a+(i-1)*h;
end
%Вычисление значение функции в узловых точках
%модифицированным методом Эйлера по формулам
%(9.13)-(9.14).
```

```
for i=2:n+1
tp=t(i-1)+h/2;
xp=x(i-1)+h/2*g(t(i-1),x(i-1));
x(i)=x(i-1)+h*g(tp,xp);
end
end
```

Листинг 9.2. Функция решения задачи Коши модифицированным методом Эйлера.

```
function [x,t]=runge_kut(a,b,n,x0)
%Функция решения задачи Коши x'(t)=g(t,x) x(a)=x0 методом
%Рунге-Кутта на интервале интегрирования [a,b],
%n - количество отрезков, на которые разбивается интервал
%[a,b].
%Вычисление шага h.
h=(b-a)/n;
x(1)=x0;
%Формирование системы равноотстоящих узлов ti
for i=1:n+1
t(i)=a+(i-1)*h;
end
%Вычисление значение функции в узловых точках методом
%Рунге-Кутта по формуле (9.19).
for i=2:n+1
% Расчет коэффициентов K1, K2, K3, K4
K1=g(t(i-1),x(i-1));
K2=g(t(i-1)+h/2,x(i-1)+h/2*K1);
K3=g(t(i-1)+h/2,x(i-1)+h/2*K2);
K4=g(t(i-1)+h,x(i-1)+h*K3);
%Расчет приращения delt
delt=h/6*(K1+2*K2+2*K3+K4);
x(i)=x(i-1)+delt;
end
end
```

Листинг 9.3. Функция решения задачи Коши методом Рунге-Кутта.

```
function [x,t,j]=kut_merson(a,b,n,eps,x0)
%Функция решения задачи Коши x'(t)=g(t,x) x(a)=x0 методом
%Кутта-Мерсона на интервале интегрирования [a,b]
%с точноттью eps, n - количество отрезков, на которые
% вначале разбивается интервал [a,b].
% Вычисление шага h.
h=(b-a)/n;
x(1)=x0;t(1)=a;i=2;
while (t(i-1)+h)<=b
R=3*eps;
while R>eps
%Расчет коэффициентов K1, K2, K3, K4, K5.
K1=g(t(i-1),x(i-1));
K2=g(t(i-1)+h/3,x(i-1)+h/3*K1);
K3=g(t(i-1)+h/3,x(i-1)+h/6*K1+h/6*K2);
K4=g(t(i-1)+h/2,x(i-1)+h/8*K1+3*h/8*K2);
K5=g(t(i-1)+h,x(i-1)+h/2*K1-3*h/2*K3+2*h*K4);
```

```
%Вычисление сравниваемых значений x(i+1)
X1=x(i-1)+h/2*(K1-3*K3+4*K4);
X2=x(i-1)+h/6*(K1+4*K4+K5);
%Вычисление оценочного коэффициента R.
R=0.2*abs(X1-X2);
%Сравнение оценочного коэффициента R с точностью eps.
if R>eps
h=h/2;
else
%Если оценочный коэффициента R меньше точности eps,
%то происходит формирование очередной найденной точки и
%переход к следующему этапу по i.
t(i)=t(i-1)+h;
x(i)=X2;
i=i+1;
%Если оценочный коэффициент R меньше eps/64,
%то можно попробовать увеличить шаг.
if R<=eps/64
if (t(i-1)+2*h)<=b
h=2*h;
end
end
end
end
end
%В переменной j возвращается количество элементов в
%массивах x и t
j=i-1
end
```

Листинг 9.4. Функция решения задачи Коши методом Кутта-Мерсона.

```
function [x,t]=adams(a,b,n,x0)
% Функция решения задачи Коши x'(t)=g(t,x) x(a)=x0 методом
%Адамса на интервале интегрирования [a,b],
%n - количество отрезков, на которые разбивается интервал
%[a,b].
% Вычисление шага h
h=(b-a)/n;
x(1)=x0;
%Формирование системы равноотстоящих узлов ti.
for i=1:n+1
t(i)=a+(i-1)*h;
end
%Вычисление значение функции в трех узловых точках методом
%Рунге-Кутта по формуле(9.19)
for i=2:4
K1=g(t(i-1),x(i-1));
K2=g(t(i-1)+h/2,x(i-1)+h/2*K1);
K3=g(t(i-1)+h/2,x(i-1)+h/2*K2);
K4=g(t(i-1)+h,x(i-1)+h*K3);
%Расчет приращения delt
delt=h/6*(K1+2*K2+2*K3+K4);
```

```
x(i)=x(i-1)+delt;
end
%Вычисление значение в остальных точках методом Адамса
for i=4:n
%Вычисление прогноза
xp=x(i)+h/24*(-9*g(t(i-3),x(i-3))+37*g(t(i-2),x(i-2))...
-59*g(t(i-1),x(i-1))+55*g(t(i),x(i)));
% Вычисление корректора
x(i+1)=x(i)+h/24*(g(t(i-2),x(i-2))-5*g(t(i-1),x(i-1))...
+19*g(t(i),x(i))+9*g(t(i+1),xp));
end
end
```

Листинг 9.5. Функция решения задачи Коши методом Адамса.

```
function [x,t]=miln(a,b,n,x0)
%Функция решения задачи Коши x'(t)=g(t,x) x(a)=x0 методом
%Милна на интервал интегрирования [a,b],n — количество
%отрезков, на которые разбивается интервал [a,b].
%Вычисление шага h
h=(b-a)/n;x(1)=x0;xp(1)=x(1);
%Формирование системы равноотстоящих узлов ti
for i=1:n+1
t(i)=a+(i-1)*h;
end
%Вычисление значение функции в трех узловых точках методом
%Рунге-Кутта по формуле(9.19)
for i=2:4
K1=g(t(i-1),x(i-1));
K2=g(t(i-1)+h/2,x(i-1)+h/2*K1);
K3=g(t(i-1)+h/2,x(i-1)+h/2*K2);
K4=g(t(i-1)+h,x(i-1)+h*K3);
%Расчет приращения delt
delt=h/6*(K1+2*K2+2*K3+K4);
x(i)=x(i-1)+delt;
xp(i)=x(i);
end
%Вычисление значение в остальных точках методом Адамса
for i=4:n
%Вычисление прогноза
xp(i+1)=x(i-3)+4*h/3*(2*g(t(i-2),x(i-2))-g(t(i-1),...
x(i-1))+g(t(i),x(i)));
%Вычисление управляющего параметра
m=xp(i+1)+28/29*(x(i)-xp(i));
%Вычисление корректора.
x(i+1)=x(i-1)+h/3*(g(t(i-1),x(i-1))+4*g(t(i),x(i))...
+g(t(i+1),m));
end
end
```

Листинг 9.6. Функция решения задачи Коши методом Милна[1].

Рассмотрим использование приведенных выше функций на примере решения следующей задачи Коши.

1 Написать функцию модифицированного метода Милна авторы предоставляют читателю.

ПРИМЕР 9.1.

Решить задачу Коши $\begin{cases} y'(x)=6y-13x^3-22x^2+17x-11+\sin(x) \\ y(0)=2 \end{cases}$.

Известно точное решение задачи 9.1

$$y(x)=\frac{119}{296}e^{6x}+\frac{1}{24}(52x^3+114x^2-30x+39)-\frac{6\sin(x)}{37}-\frac{\cos(x)}{37}.$$

На листинге 9.7 представлено решение уравнения методами:

- модифицированным методом Эйлера;
- Рунге-Кутта;
- Кутта-Мерсона;
- Адамса;
- Милна.

```
%Точное решение
function q=fi(x)
q=119/296*exp(6*x)+1/24*(52*x.^3+114*x.^2-30*x+39)-...
6*sin(x)/37-cos(x)/37;
end
%Правая часть дифференциального уравнения.
function y=g(t,x)
y=6*x-13*t^3-22*t^2+17*t-11+sin(t);
end
%Функция решения задачи Коши модифицированным методом
%Эйлера.
function [x,t]=eiler_m(a,b,n,x0)
h=(b-a)/n;
x(1)=x0;
for i=1:n+1
t(i)=a+(i-1)*h;
end
for i=2:n+1
tp=t(i-1)+h/2;
xp=x(i-1)+h/2*g(t(i-1),x(i-1));
x(i)=x(i-1)+h*g(tp,xp);
end
end
%Функция решения задачи Коши методом Рунге-Кутта.
function [x,t]=runge_kut(a,b,n,x0)
h=(b-a)/n;
x(1)=x0;
for i=1:n+1
t(i)=a+(i-1)*h;
end
for i=2:n+1
K1=g(t(i-1),x(i-1));
K2=g(t(i-1)+h/2,x(i-1)+h/2*K1);
K3=g(t(i-1)+h/2,x(i-1)+h/2*K2);
K4=g(t(i-1)+h,x(i-1)+h*K3);
delt=h/6*(K1+2*K2+2*K3+K4);
x(i)=x(i-1)+delt;
end
```

```
end
%Функция решения задачи Коши методом Кутта-Мерсона.
function [x,t,j]=kut_merson(a,b,n,eps,x0)
h=(b-a)/n;
x(1)=x0;t(1)=a;i=2;
while (t(i-1)+h)<=b
R=3*eps;
while R>eps
K1=g(t(i-1),x(i-1));
K2=g(t(i-1)+h/3,x(i-1)+h/3*K1);
K3=g(t(i-1)+h/3,x(i-1)+h/6*K1+h/6*K2);
K4=g(t(i-1)+h/2,x(i-1)+h/8*K1+3*h/8*K2);
K5=g(t(i-1)+h,x(i-1)+h/2*K1-3*h/2*K3+2*h*K4);
X1=x(i-1)+h/2*(K1-3*K3+4*K4);
X2=x(i-1)+h/6*(K1+4*K4+K5);
R=0.2*abs(X1-X2);
if R>eps
h=h/2;
else
t(i)=t(i-1)+h;
x(i)=X2;
i=i+1;
if R<=eps/64
if (t(i-1)+2*h)<=b
h=2*h;
end
end
end
end
end
end
j=i-1
end
%Функция решения задачи Коши методом Милна.
function [x,t]=miln(a,b,n,x0)
h=(b-a)/n;
x(1)=x0;xp(1)=x(1);
for i=1:n+1
t(i)=a+(i-1)*h;
end
for i=2:4
K1=g(t(i-1),x(i-1));
K2=g(t(i-1)+h/2,x(i-1)+h/2*K1);
K3=g(t(i-1)+h/2,x(i-1)+h/2*K2);
K4=g(t(i-1)+h,x(i-1)+h*K3);
delt=h/6*(K1+2*K2+2*K3+K4);
x(i)=x(i-1)+delt;
xp(i)=x(i);
end
for i=4:n
xp(i+1)=x(i-3)+4*h/3*(2*g(t(i-2),x(i-2))-g(t(i-1),...
x(i-1))+g(t(i),x(i)));
```

```
m=xp(i+1)+28/29*(x(i)-xp(i));
x(i+1)=x(i-1)+h/3*(g(t(i-1),x(i-1))+4*g(t(i),x(i))...
+g(t(i+1),m));
end
end
%Функция решения задачи Коши методом Адамса.
function [x,t]=adams(a,b,n,x0)
h=(b-a)/n;
x(1)=x0;
for i=1:n+1
t(i)=a+(i-1)*h;
end
for i=2:4
K1=g(t(i-1),x(i-1));K2=g(t(i-1)+h/2,x(i-1)+h/2*K1);
K3=g(t(i-1)+h/2,x(i-1)+h/2*K2);K4=g(t(i-1)+h,x(i-1)+h*K3);
delt=h/6*(K1+2*K2+2*K3+K4);
x(i)=x(i-1)+delt;
end
for i=4:n
xp=x(i)+h/24*(-9*g(t(i-3),x(i-3))+37*g(t(i-2),x(i-2))...
-59*g(t(i-1),x(i-1))+55*g(t(i),x(i)));
x(i+1)=x(i)+h/24*(g(t(i-2),x(i-2))-5*g(t(i-1),x(i-1))...
+19*g(t(i),x(i))+9*g(t(i+1),xp));
end end
%Решение дифференциального уравнения модифицированным
% методом Эйлера.
[YE_M,XE_M]=eiler_m(0,1,10,2);
%Решение дифференциального уравнения методом Рунге-Кутта.
[YR,XR]=runge_kut(0,1,10,2);
%Решение дифференциального уравнения методом
%Кутта-Мерсона.
[YKM,XKM,KM]=kut_merson(0,1,5,0.001,2);
%Решение дифференциального уравнения методом Адамса.
[YA,XA]=adams(0,1,10,2);
%Решение дифференциального уравнения методом Милна.
[YM,XM]=miln(0,1,10,2);
%Точное решение.
x1=0:0.05:1;y1=fi(x1);
%Построение графиков.
plot(x1,y1,'-g;exact solution;',XE_M,YE_M,'*b;eiler;',...
XR,YR,'ob;runge-kutt;',XA,YA,'^b;adams;',...
XM,YM,'>b;miln;');
figure();
plot(x1,y1,'-g;exact solution;',XKM,YKM,'<b;kut-merson;');
grid on;
```

Листинг 9.7.

На рис. 9.3-9.4 приведены графики решения задачи модифицированным методом Эйлера, методами Рунге-Кутта, Кутта-Мерсона, Адамса, Милна и точного решения. При обращении к функции kut_merson в качестве *n* передавалось число 5.

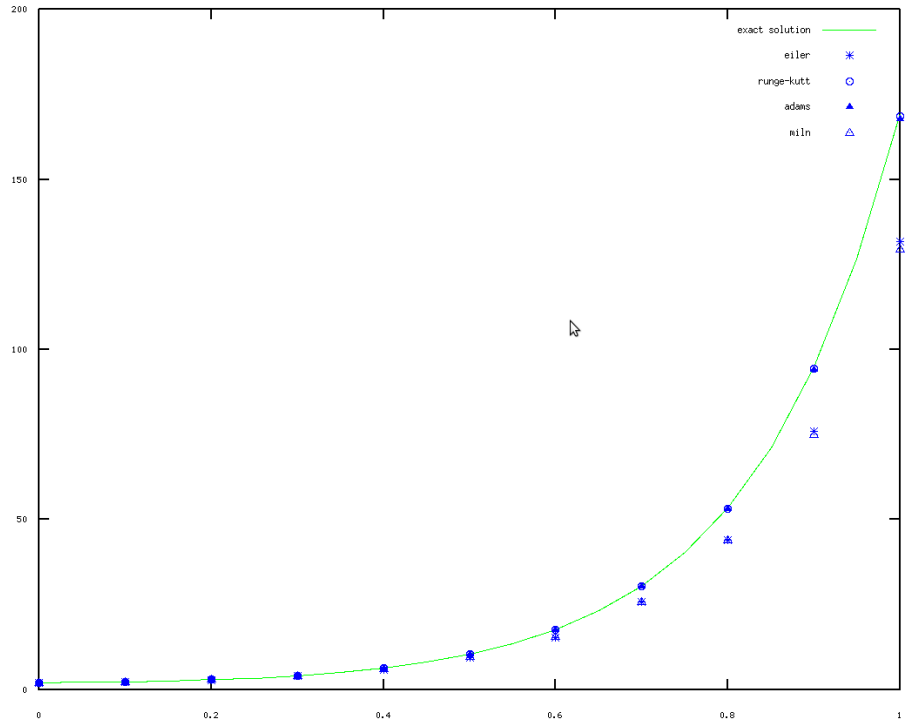

Рисунок *9.3* Графики решения модифицированным методом Эйлера, методами Рунге-Кутта, Адамса, Милна и точного решения

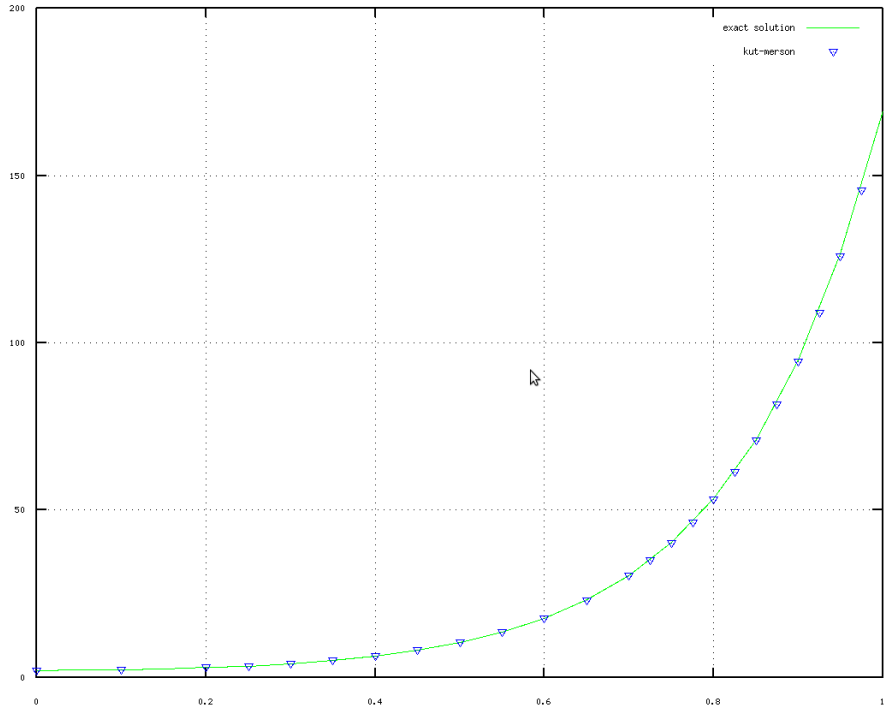

Рисунок *9.4* Графики решения методом Кутта-Мерсона и точного решения

Функция выбирала оптимальный шаг на каждом из отрезков. Как видно из рисунка, шаг то увеличивается, то уменьшается. При решении уравнения методом Кутта-Мерсона невозможно гарантировать вычисление значения в заданных точках интервала. Однако после получения решения методом Кутта-Мерсона значение в любой точки можно вычислить, интерполируя полученную зависимость.

При решении данной задачи наиболее точными оказались методы Адамса, Рунге-Кутта и Кутта-Мерсона.

9.4 Решение систем дифференциальных уравнений

Все рассмотренные методы решения дифференциальных уравнений применимы и для систем дифференциальных уравнений. Рассмотрим на примере метода Рунге-Кутта, как рассмотренные методы можно обобщить для систем.

Пусть дана система дифференциальных уравнений в матричном виде

$$\frac{d\bar{x}}{dt}=\bar{f}(t,\bar{x}) \tag{9.30}$$

с начальным условием $\bar{x}(t_0)=\bar{x}_0$,

где $\bar{x}=\begin{pmatrix}x_1(t)\\x_2(t)\\...\\x_n(t)\end{pmatrix}$, $\bar{f}(t,\bar{x})=\begin{pmatrix}f_1(t,x_1,x_{2,}...,X_n)\\f_2(t,x_1,x_{2,}...,X_n)\\...\\f_n(t,x_1,x_{2,}...,X_n)\end{pmatrix}$, $\bar{x^0}=\begin{pmatrix}x_1^0\\x_2^0\\...\\x_n^0\end{pmatrix}$.

Задавшись некоторым шагом h и введя стандартные обозначения $t_i=t_0+i\,h$, $x_i=x(t_i)$, $\Delta x_i=x_{i+1}-x_i$, $i=1,n$ получим формулы метода Рунге-Кутта для системы:

$$\begin{cases}\bar{x_{i+1}}=\bar{x}_i+\Delta\bar{x}_i,\ i=1,n\\ \Delta\bar{x}_i=\frac{h}{6}(\bar{K}_1^i+2\bar{K}_2^i+2\bar{K}_3^i+\bar{K}_4^i)\\ \bar{K}_1^i=\bar{f}(t_i,\bar{x}_i)\\ \bar{K}_2^i=\bar{f}(t_i+\frac{h}{2},\bar{x}_i+\frac{h}{2}\bar{K}_1^i)\\ \bar{K}_3^i=\bar{f}(t_i+\frac{h}{2},\bar{x}_i+\frac{h}{2}\bar{K}_2^i)\\ \bar{K}_4^i=\bar{f}(t_i+h,\bar{x}_i+h\bar{K}_3^i)\end{cases} \tag{9.31}$$

9.5 Встроенные функции ***Octave*** решения для дифференциальных уравнений

Наиболее часто используемыми в Octave функциями для решения дифференциальных уравнений являются:

- `ode23(@f, interval, X0, options)`, `ode45(@f, interval, X0, options)` – функции решений обыкновенных нежёстких дифференциальных уравнений (или систем) методом Рунге-Кутта 2-3-го и 4-5-го порядка точности соответсвенно;
- `ode5r(f, interval, X0, options)`, `ode2r(f, interval, X0,`

`options)` – – функции решений обыкновенных жёстких дифференциальных уравнений (или систем).

Функции решают систему дифференциальные уравнения (системы), автоматически подбирая шаг для достижения необходимой точности.

Входными параметрами этих функций являются:

- `f` – вектор-функция для вычисления правой части дифференциального уравнения или системы;
- `interval` – массив из двух чисел, определяющий интервал интегрирования дифференциального уравнения или системы;
- `X0` – вектор начальных условий системы дифференциальных систем;
- `options` – параметры управления ходом решения дифференциального уравнения или системы.

Для определения параметров управления ходом решения дифференциальных уравнений используется функция `odeset` следующей структуры

```
options = odeset ("namepar1", val1, "namepar2", val2, …, "nameparn", valn);
```

Здесь

- `"namepari"` – имя i-го параметра;
- `vali` – значение i-го параметра.

При решении дифференциальных уравнений необходимо определить следующие параметры:

- `RelTol` – относительная точность решения, значение по умолчанию 10^{-3} ;
- `AbsTol` – абсолютная точность решения, значение по умолчанию 10^{-3} ;
- `InitialStep` – начальное значение шага изменения независимой переменной, значение по умолчанию 0.025;
- `MaxStep` – максимальное значение шага изменения независимой переменной, значение по умолчанию 0.025.

Все функции возвращают:

- массив `T` – координат узлов сетки, в которых ищется решение;
- матрицу `X`, *i*-й столбец которой является значением вектор-функции решения в узле T_i .

Напомним читателю определение жёсткой системы дифференциальных уравнений. Система дифференциальных уравнений n-го порядка

$$\frac{d\,\boldsymbol{x}}{dt} = \boldsymbol{B}\,x \tag{9.32}$$

называется жёсткой [2], если выполнены следующие условия:

- действительные части всех собственных чисел матрицы $\boldsymbol{B}(n)$ отрицательны $\left|\mathrm{Re}(\lambda_k)\right| < 0, k = 1,2,\ldots,n$;
- величина $s = \dfrac{\max\limits_{1 \le k \le n} \left|\mathrm{Re}(\lambda_k)\right|}{\min\limits_{1 \le k \le n} \left|\mathrm{Re}(\lambda_k)\right|}$, называемая числом жесткости системы, велика.

При исследовании на жёсткость нелинейной системы дифференциальных уравнений (9.30) в роли матрицы **B** будет выступать матрица частных производных.

Решим задачу 9.1 с использованием функций `ode23`, `ode45`. Текст программы с комментариями представлен на листинге 9.8.

```
%Точное решение системы.
function q=fi(x)
q=119/296*exp(6*x)+1/24*(52*x.^3+114*x.^2-30*x+39)-...
6*sin(x)/37-cos(x)/37;
end
%Правая часть дифференциального уравнения.
function y=g(t,x)
y=6*x-13*t^3-22*t^2+17*t-11+sin(t);
end
%Определение параметров управления ходом решения
% дифференциального уравнения.
%RelTol – относительная точность решения 1E-5,
%AbsTol – абсолютная точность решения 1E-5,
%InitialStep – начальное значение шага изменения
%независимой переменной 0.025,
%MaxStep – максимальное значение шага изменения
%независимой переменной 0.1.
par=odeset ("RelTol", 1e-5, "AbsTol", 1e-5,...
'InitialStep',0.025,'MaxStep',0.1);
%Решение дифференциального уравнения методом Рунге-Кутта
%2-3 порядка.
[X23,Y23]=ode23(@g,[0 0.25],2,par);
%Определение параметров управления ходом решения
% дифференциального уравнения.
%RelTol – относительная точность решения 1E-4,
%AbsTol – абсолютная точность решения 1E-4,
%InitialStep – начальное значение шага изменения
%независимой переменной 0.005,
%MaxStep – максимальное значение шага изменения
%независимой переменной 0.2.
par=odeset ("RelTol", 1e-4, "AbsTol",...
1e-4,'InitialStep',0.05,'MaxStep',0.2);
%Решение дифференциального уравнения методом Рунге-Кутта
%4-5 порядка.
[X45,Y45]=ode45(@g,[0 0.25],2,par);
%Точное решение
x1=0:0.05:0.25;
y1=fi(x1)
%График решения функцией ode23 и точного решения.
plot(x1,y1,'-g;exact solution;',X23,Y23,'*b;ode23;');
grid on;
figure();
%График решения функцией ode45 и точного решения.
plot(x1,y1,'-g;exact solution;',X45,Y45,'*b;ode45;');
grid on;
```

Листинг 9.8.

На рис. 9.5 представлено решение, найденное с помощью функции `ode23` с точностью 1E-5 и точное решение. На рис. 9.6 представлено решение, найденное с помощью функции `ode45` с точностью 1E-4 и точное решение.

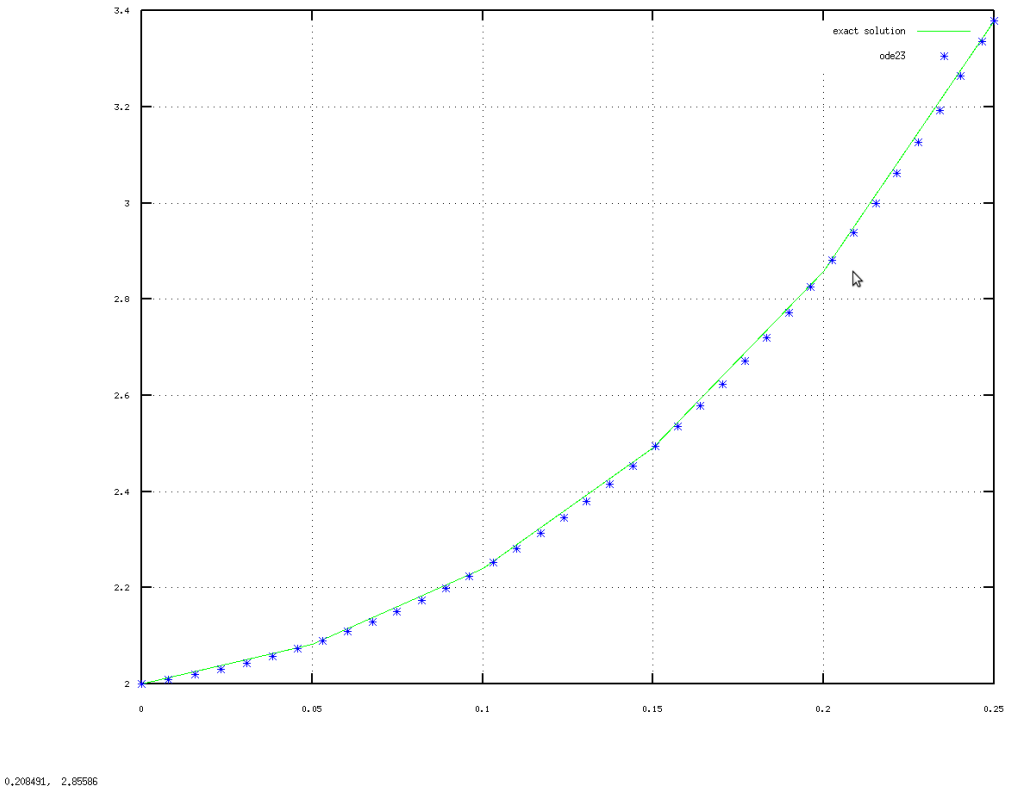

Рисунок *9.5*. Графики точного решения задачи *9.1* и решения, найденного с помощью функции *ode23*

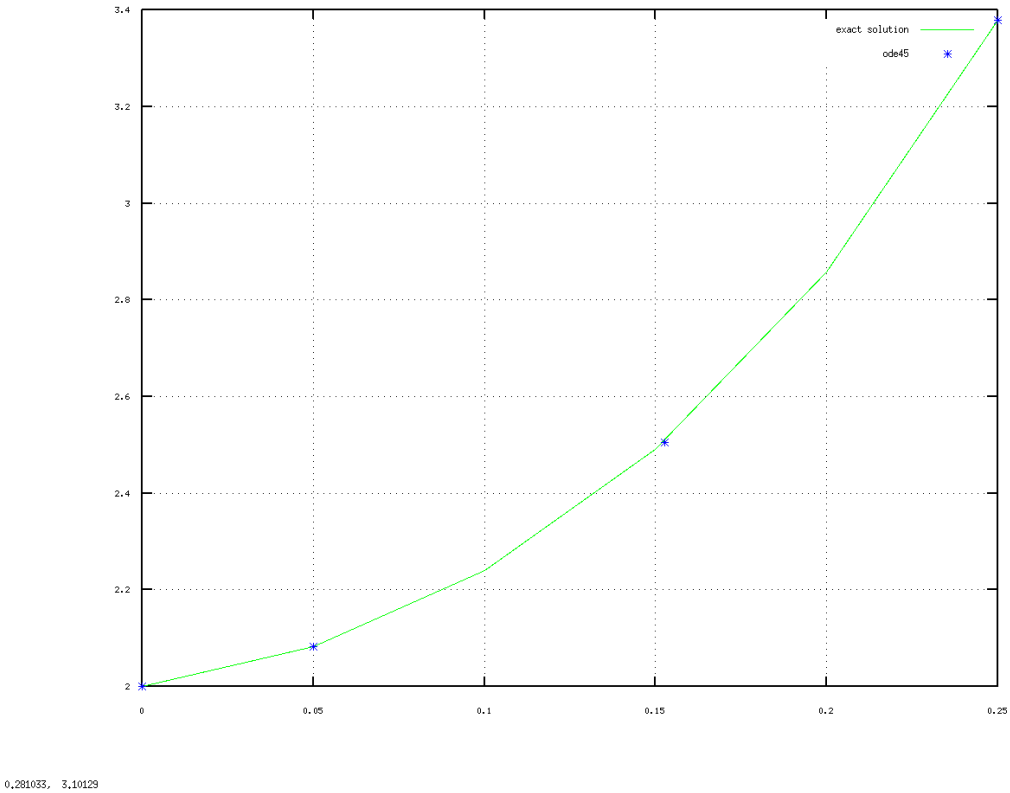

Рисунок *9.6* Графики точного решения задачи *9.1* и решения, найденного с помощью функции *ode45*

Функции ode23 и ode45 позволяют найти решение с заданной точностью, однако, как и следовало ожидать, при использовании метода Рунге-Кутта более высокой точности шаг изменения переменной x намного меньше.

Пример решения жёсткой системы дифференциальных уравнений

ПРИМЕР 9.2.

Решить задачу Коши для жёсткой системы дифференциальных уравнений

$$\frac{d\boldsymbol{x}}{dt}=\begin{pmatrix}119.46 & 185.38 & 126.88 & 121.03\\ -10.395 & -10.136 & -3.636 & 8.577\\ -53.302 & -85.932 & -63.182 & -54.211\\ -115.58 & -181.75 & -112.8 & -199\end{pmatrix}\boldsymbol{x}, \qquad \boldsymbol{x}(0)=\begin{pmatrix}1\\1\\1\\1\end{pmatrix}.$$

Решение задачи с комментариями представлено на листинге, на рис. Можно увидеть график решения.

```
%Функция правой части жёсткой системы дифференциальных
% уравнений.
function dx=syst1(t,x)
B=[119.46 185.38 126.88 121.03;-10.395 -10.136 -3.636...
 8.577; -53.302 -85.932 -63.182 -54.211;-115.58 -181.75...
 -112.8 -199];
dx=B*x;
end
%Определение параметров управления ходом решения жёсткой
% системы дифференциальных уравнений.
%RelTol – относительная точность решения 1E-8,
%AbsTol – абсолютная точность решения 1E-8,
%InitialStep – начальное значение шага изменения
%независимой переменной 0.02,
%MaxStep – максимальное значение шага изменения
%независимой переменной 0.1.
par=odeset ("RelTol", 1e-8, "AbsTol", 1e-8,...
'InitialStep',0.02,'MaxStep',0.1);
%Решение жёсткой системы дифференциальных уравнений.
%Построение графика решения.
[A,B]=ode2r(@syst1,[0 5],[1;1;1;1]);
plot(A,B,'-k')
grid on;
```

Листинг 9.9.

Этим примером мы заканчиваем краткое описание возможностей Octave для решения дифференциальных уравнений. Однако, следует помнить о следующем: решение реального дифференциального уравнения (а тем более системы) – достаточно сложная математическая задача. Для её решения недостаточно знания синтаксиса функций Octave, необходимо достаточно глубоко знать математические методы решения подобных задач. При решении дифференциальных уравнений необходимо определить метод решения и только потом пытаться использовать встроенные функции или писать свои. Авторы не случайно достаточно подробно напомнили читателю основные численные методы решения дифференциальных уравнений и систем. На наш взгляд без знания численных и аналитических методов решения дифференциальных уравнений, достаточно проблематично решить реальную задачу.

Кроме того, следует помнить, что функциями ode23, ode45, ode2r, ode5r возможности пакета не ограничиваются. Octave предоставляет достаточное количество функций для решения дифференциальных уравнений различного вида. Они подробно

описаны в справке консольной версии приложения[2].

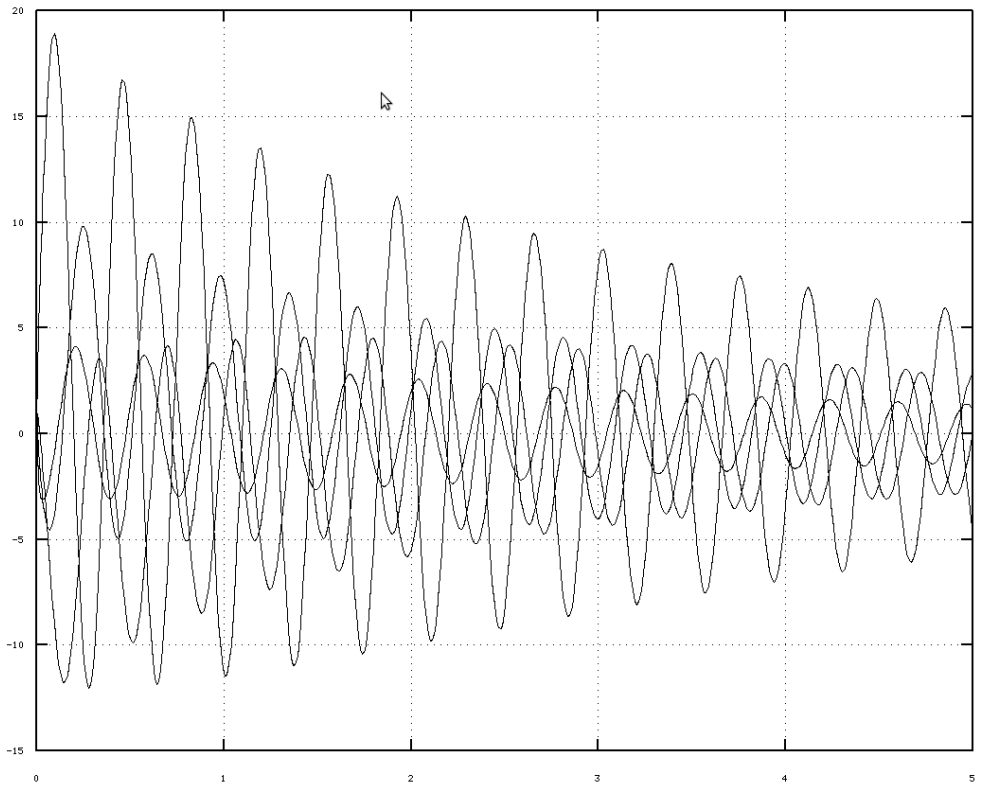

Рисунок *9.7* График решения задачи *9.2*

Множество функций решени дифференциальных уравнений находится в пакете расширений *odepkg*. Краткое описание функций этого пакета на английском языке с некоторыми примерами приведено на странице http://octave.sourceforge.net/odepkg/overview.html.

2 Ещё раз напоминаем читателю, что справка по Octave, доступная из оболочки qtoctave недостаточно полная.