ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ФИЗИЧЕСКИХ ПРОЦЕССОВ
ЛЕКЦИЯ №9
«Многошаговые методы решения дифференциальных уравнений»

Отделение ядерно-топливного цикла

Лектор:
Зав. каф. - руководитель ОЯТЦ ИЯТШ
Горюнов А.Г.

2020

План лекции

9.1 Многошаговые методы решения задачи Коши

9.2 Проблема численной устойчивости

9.3 Методы решения дифференциальных уравнений в Matab

9.4 Моделирование динамических систем, описываемых системами ОДУ в Simulink

Информация по курсу:

https://portal.tpu.ru/SHARED/a/ALEX1479/study/Matmod/Tab

9.1 Многошаговые методы решения задачи Коши

9.1.1 Общая схема многошаговых методов

В одношаговых методах решения задачи Коши для одного уравнения:

$$y' = f(x,y), \quad x \in [a,b]$$
$$y(a) = y_0, \tag{9.1}$$

значение y_{i+1} зависит только от информации о решении в предыдущей точке сетки x_i, $i = 0,1,2, \ldots$

Принято считать, что можно добиться повышения точности, если использовать информацию о решении в нескольких предыдущих точках сетки x_i, x_{i-1}, x_{i-2}, … . Также бывает, что целесообразным привлекать информацию с забеганием вперед за точку x_{i+1}. Подобные методы называются **многошаговыми** и могут быть получены несколькими способами.

Общая схема многошаговых методов имеет вид:

$$y_{n+k} = F\left(f, x_{n+k}, x_n, y_{n+k}, y_{n+k-1}, \ldots, y_n\right), \tag{9.2}$$

где значение приближенного решения y_{n+k} в точке x_{n+k} определяется через значения решения в k точках, предшествующих x_{n+k}.

9.1.1 Общая схема многошаговых методов

Метод (9.2) называется k - шаговым. Из класса методов (9.2) можно выделить класс **линейных многошаговых методов**:

$$\sum_{i=0}^{k} \alpha_i y_{n+i} = h \sum_{i=0}^{n} \beta_i f\left(x_{n+i}, y\left(x_{n+i}\right)\right), \quad n = 0,1,2,\ldots, \tag{9.3}$$

где α_i и β_i – постоянные, $\alpha_k \neq 0$, $|\alpha_0| + |\beta_0| \neq 0$. Равенство (9.3) устанавливает линейное соотношение между y_{n+i} и $f(x_{n+i}, y(x_{n+i}))$.

9.1.2 Методы прогноза и коррекции.

Для того чтобы вычислить последовательность приближённых значений y_n, необходимо сначала получить k начальных значений $y_0, y_1, \ldots, y_{k-1}$. При $\beta_k = 0$, метод (9.3) называется **явным линейным *k*-шаговым**, причем приближение к решению y_{n+k} вычисляется просто. В случае, если $\beta_k \neq 0$ — метод называется **неявным линейным *k*-шаговым**, т.к. правая часть (9.3) содержит $f(x_{n+k}, y(x_{n+k}))$ и в общем случае необходимо решить нелинейное уравнение относительно $y(x_{n+k})$. Для реализации неявного метода можно равенство (9.3) представить в виде:

$$y_{n+i} = h\frac{\beta_k}{\alpha_k} f\left(x_{n+k}, y\left(x_{n+k}\right)\right) + g_n, \quad \beta_k \neq 0, \tag{9.4}$$

где g_n содержит известные величины $f(x_{n+i}, y(x_{n+i}))$, $y(x_{n+i})$, $i = 0, 1, \ldots, k-1$.

9.1.2 Методы прогноза и коррекции

Доказано, что если:

$$h < \left|\frac{\alpha_k}{\beta_k}\right| \cdot \frac{1}{L}, \qquad (9.5)$$

(L – константа Липшица*), то существует единственное решение y_{n+k} уравнения (9.4), которое можно получить с помощью итерационного процесса.

*Функция называется функцией Липшица в ограниченной области Ω, если она удовлетворяет условию Липшица. Условие Липшица состоит в том, что должно выполняться неравенство:

$$\left|f(x) - f(y)\right| \le L\left\|x - y\right\| \quad \text{для всех } x, y \in \Omega, \left\|x\right\| \text{ - норма},$$

L - константа, которую называют константой Липшица.

Решение уравнения (9.4) можно получить с помощью итерационного процесса:

$$y_{n+k}^{\xi+1} = h\frac{\beta_k}{\alpha_k} f\left(x_{n+k}, y_{n+k}^{\xi}\right) + g_n, \quad \xi = 0,1,\ldots, \qquad (9.6)$$

где y_{n+k}^{0} произвольно. Выбор y_{n+k}^{0} играет огромную роль в скорости сходимости итерационного процесса, поэтому для его вычисления используется явный многошаговый метод – **предсказывающий (предиктор)**, неявный метод (9.6) в этой сцепке – **исправляющий (корректор)**.

9.1.3 Получение многошаговых методов

Многошаговые методы могут быть построены следующим образом. Исходное уравнение (9.1) для задачи Коши запишем в виде $dy(x) = f(x, y)dx$. Проинтегрируем обе части этого соотношения на отрезке $[x_i, x_{i+1}]$. Из левой части получаем:

$$\int_{x_i}^{x_{i+1}} dy(x) = y(x_{i+1}) - y(x_i) \approx y_{i+1} - y_i, \qquad (9.7)$$

где y_{i+1}, y_i – сеточные значения искомой функции.

Для вычисления интегралов правой части сначала построим интерполяционный многочлен $P_{k-1}(x)$ степени $(k - 1)$ для функции $f(x, y)$ на этом отрезке по значениям $f(x_{i-k+1}, y_{i-k+1})$, $f(x_{i-k+2}, y_{i-k+2})$, ..., $f(x_i, y_i)$. Тогда интеграл правой части примет вид:

$$\int_{x_i}^{x_{i+1}} f(x, y)dx \approx \int_{x_i}^{x_{i+1}} P_{k-1}(x)dx. \qquad (9.8)$$

Приравнивая (9.7) и (9.8), получаем формулу для определения неизвестного значения сеточной функции y_{i+1} в узле x_{i+1}:

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} P_{k-1}(x)dx. \qquad (9.9)$$

На основе (9.9) можно строить различные многошаговые методы любого порядка точности. Порядок точности при этом зависит от степени $P_{k-1}(x)$, для построения которого используются значения сеточной функции y_i, y_{i-1}, ..., y_{i-k+1}, вычисленные на k предыдущих узлах.

9.1.4 Семейство методов Адамса

Известны методы Адамса *k*-го порядка. Простейший из них при $k = 1$ повторяет метод Эйлера первого порядка точности. Метод четвертого порядка принято называть **методом Адамса**. Формулу для него получают следующим образом.

Должны быть известны в четырех последовательных узлах ($k = 4$) значения сеточной функции y_{i-3}, y_{i-2}, y_{i-1}, y_i и вычисленные первоначально значения правой части (9.1) f_{i-3}, f_{i-2}, f_{i-1}, f_i. В качестве интерполяционного многочлена $P_3(x)$ применяют многочлен Ньютона. В случае $h = \text{const}$ конечные разности для правой части в узле x_i будут иметь вид:

$$\Delta f_i = f_i - f_{i-1};$$

$$\Delta^2 f_i = f_i - 2 f_{i-1} + f_{i-2};$$

$$\Delta^3 f_i = f_i - 3 f_{i-1} + 3 f_{i-2} - f_{i-3}.$$

Тогда разностная схема метода Адамса запишется в виде:

$$y_{i+1} = y_i + h \cdot f_i + \frac{h^2}{2}\Delta f_i + \frac{5h^3}{12}\Delta^2 f_i + \frac{3h^4}{8}\Delta^3 f_i. \qquad (9.10)$$

По сравнению с методом Рунге - Кутта той же точности можно отметить его экономичность, так как (9.10) требует на каждом шаге только один раз вычисление правой части в соотношении (9.1). Однако расчет здесь можно начать только с узла x_4. Значения y_1, y_2, y_3, необходимые для вычисления y_4, необходимо определять одношаговым методом, что усложняет алгоритм вычисления. Кроме того, метод Адамса не позволяет изменять шаг *h* в процессе моделирования, что доступно для одношаговых методов.

9.1.5 Неявные многошаговые методы

В литературе неявные многошаговые методы называются методами **прогноза и коррекции** или (**методами предиктор - корректор**).

Принцип состоит в том, что на каждом шаге расчета вводятся два этапа, использующие многошаговые методы:
1) с помощью явного метода (**предиктора**) по известным значениям функции в предыдущих узлах находится начальное значение

$$y_{i+1} = y_{i+1}^{(0)}$$

в новом узле;

2) используя неявный метод (**корректор**) в результате итераций вычисляют приближения

$$y_{i+1}^{(1)}, y_{i+1}^{(2)}, \ldots, y_{i+1}^{(k)}, y_{i+1}^{(k+1)}, \ldots$$

Посредством корректора итерации продолжаются до тех пор, пока не будет достигнута желаемая точность, далее осуществляется переход к следующей точке сетки, т. е. по рассмотренному выше алгоритму определяется значение y_{i+2}.

9.1.5 Неявные многошаговые методы

Одним из вариантов метода прогноза и коррекции является метод на основе метода Адамса четвертого порядка.

Вид разностных соотношений **на этапе предиктора**:

$$y_{i+1} = y_i + \frac{h}{24}\left(55 f_i + 59 f_{i-1} + 37 f_{i-2} - 9 f_{i-3}\right); \qquad (9.11)$$

на этапе корректора:

$$y_{i+1} = y_i + \frac{h}{24}\left(9 f_{i+1} + 19 f_i - 5 f_{i-1} + f_{i-2}\right). \qquad (9.12)$$

В (9.11) и (.12) используются значения правой части (9.1), а не значения конечных разностей. Явная схема (9.11) используется на каждом шаге лишь один раз, а с помощью неявной схемы (9.12) выполняется итерационный процесс вычислений y_{i+1}, поскольку это значение входит в правую часть выражения $f_{i+1} = f(x_{i+1}, y_{i+1})$.

В формулах (9.11) и (9.12), как и в случае метода Адамса, при вычислении y_{i+1} необходимы значения сеточной функции в четырех предыдущих узлах: y_{i-3}, y_{i-2}, y_{i-1}, y_i.

Расчет по этому методу может быть начат только со значения y_4. Необходимые при этом значения y_1, y_2 и y_3 вычисляются по методу Рунге - Кутта, y_0 задается начальным условием.

Метод Адамса легко распространяется на системы дифференциальных уравнений.

9.1.6 Повышение точности неявных методов

Точность можно повысить путем уменьшения значения шага *h*. Однако этот путь ограничен требованием экономичности, поскольку это требует существенного увеличения объема вычислений.

На практике часто для повышения точности численного решения без существенного увеличения машинного времени используется **метод Рунге**.

Суть **метода Рунге** состоит в том, что по одной и той же разностной схеме проводятся повторные расчеты с различными шагами. В соответствие с методом Рунге уточненное значение сеточной функции

$$y_h^*$$

в узлах сетки с шагом *h* вычисляется по формуле:

$$y_h^* = \frac{2^k y_{h/2} - y_h}{2^k - 1} + O\left(h^{k+1}\right).$$

Порядок точности этого решения равен (*k* + 1), при этом, используемая разностная схема имеет порядок точности *k*, т. е. точность **повышается на порядок**.

9.2 Проблема численной устойчивости

Введем полином вида:

$$\rho(\theta) = \sum_{i=0}^{k} \alpha_i \theta^i \qquad (9.13)$$

будем в дальнейшем называть его **характеристическим**.

Для неявного метода

$$y_{n+i} = h\frac{\beta_k}{\alpha_k} f\left(x_{n+k}, y\left(x_{n+k}\right)\right) + g_n, \quad \beta_k \neq 0, \qquad (9.4)$$

полная погрешность в узле x_n определяется разностью $y_n - y(x_n)$, $0 \le n \le N$. Если она стремиться к нулю при $h \to 0$, то **метод сходится**.

Метод из класса (9.3) **сходится**, если для каждой задачи (9.1)

$$y_n \to y(x) \text{ при } h \to 0, \; n = \frac{x - x_0}{h}$$

для любых $x \in [x_0, x_k]$.

Каким условиям должен удовлетворять метод из класса (9.3)?

Прежде всего в качестве такого условия должен выступать минимальный уровень локальной точности.

9.2 Проблема численной устойчивости

Невязка ρ_{n+k} , которая получается после подстановки точного решения $y(x)$ дифференциального уравнения в разностное (9.3),

$$\rho_{n+k} = \sum_{i=0}^{k} \alpha_i y_{n+i} - h \sum_{i=0}^{n} \beta_i f\left(x_{n+i}, y\left(x_{n+1}\right)\right), \qquad (9.14)$$

имеет порядок $O(h^{s+1})$ и называется погрешностью аппроксимации. Число s называется порядком аппроксимации или **степенью разностного уравнения** (9.3), а $r_{n+k} = (\rho_{n+k})/h$ – погрешностью дискретизации.

Метод (9.3) является **согласованным**, если

$$\max_{0 \le n \le N} \left\| r_{n+k} \right\| \to 0 \text{ при } h \to 0$$

и имеет порядок согласованности s ,

$$\max_{0 \le n \le N} \left\| r_{n+k} \right\| = O\left(h^s\right)$$

9.2 Проблема численной устойчивости

Метод из класса (9.3) удовлетворяет **корневому условию**, если все корни характеристического полинома $\rho(\theta)$ лежат внутри единичной окружности или на самой окружности, причем те корни, которые лежат на единичной окружности, являются простыми.

Если метод **согласован**, то $\rho(\theta)$ обязательно имеет корень $\theta_1 = +1$.

Корни характеристического полинома классифицируются следующим образом:

$\theta_1 = +1$ – главный корень;

$|\theta_i| \le 1$, $i = 2, 3, \ldots, k$ – посторонние корни.

Метод удовлетворяющий корневому условию называют **нуль-устойчивым**.

Согласованность – определяет величину погрешности аппроксимации, **нуль-устойчивость** – определяет характер развития этой и других погрешностей в пределе при $h \to 0$, $Nh = x_k - x_0$.

Метод из класса (9.3) **сходится** тогда и только тогда, когда он является **согласованным** и **нуль-устойчивым**.

9.3 Методы решения дифференциальных уравнений в Matab

Для решения систем ОДУ:

$$\frac{d\vec{Y}}{dt} = \vec{F}\left(\vec{Y}, t\right)$$
$$\vec{Y}\left(t_0\right) = \vec{Y}_0 \qquad (9.15)$$

в MatLAB реализованы различные методы. Их реализации названы **решателями** ОДУ. Решатели реализуют следующие методы решения систем дифференциальных уравнений:

9.3.1 Нежесткие решатели

ode45	Решение нежестких дифференциальных уравнений — метод среднего порядка точности
ode23	Решение нежестких дифференциальных уравнений — метод низкого порядка точности
ode113	Решение нежестких дифференциальных уравнений — метод переменного порядка точности

9.3 Методы решения дифференциальных уравнений в Matab

Для решения систем ОДУ:

$$\frac{d\vec{Y}}{dt} = \vec{F}\left(\vec{Y}, t\right)$$
$$\vec{Y}\left(t_0\right) = \vec{Y}_0 \qquad (9.15)$$

в MatLAB реализованы различные методы. Их реализации названы **решателями** ОДУ. Решатели реализуют следующие методы решения систем дифференциальных уравнений:

9.3.1 Нежесткие решатели

ode45	Решение нежестких дифференциальных уравнений — метод среднего порядка точности
ode23	Решение нежестких дифференциальных уравнений — метод низкого порядка точности
ode113	Решение нежестких дифференциальных уравнений — метод переменного порядка точности

9.3.1 Нежесткие решатели

ode45 – одношаговые явные методы Рунге-Кутта 4-го и 5-го порядка {начальное пробное решение}. Во многих случаях он дает хорошие результаты;

ode23 – одношаговые явные методы Рунге-Кутта 2-го и 3-го порядка. При умеренной жесткости системы ОДУ и низких требованиях к точности этот метод может дать выигрыш в скорости решения;

ode113 – многошаговый метод Адамса-Башворта-Мултона переменного порядка. Это адаптивный метод, который может обеспечить высокую точность решения.

9.3.2 Жесткие решатели

ode15s	Решение жестких дифференциальных уравнений и ДАУ — метод переменного порядка точности
ode23s	Решите жесткие дифференциальные уравнения — метод низкого порядка точности
ode23t	Решение умеренно жестких ОДУ и ДАУ — метод трапеций
ode23tb	Решение жестких дифференциальных уравнений — метод трапеций + формула дифференцирования назад

ode15s – многошаговый метод переменного порядка (от 1-го до 5-го, по умолчанию 5), использующий формулы численного дифференцирования. Это адаптивный метод, его стоит применять, если решатель ode45 не обеспечивает решения;

ode23s – одношаговый метод, использующий модифицированную формулу Розенброка2-го порядка. Может обеспечить высокую скорость вычислений при низкой точности;

ode23t – метод трапеций с интерполяцией. Этот метод дает хорошие результаты прирешении задач, описывающих осцилляторы с почти гармоническим выходным сигналом;

ode23tb – неявный метод Рунге-Кутта в начале решения и метод, использующий формулы обратного дифференцирования 2-го порядка в последующем. При низкой точности этот метод может оказаться более эффективным, чем ode15s.

9.3.3 Синтаксис ODE на примере ode45 (https://docs.exponenta.ru/matlab/ref/ode45.html)

Синтаксис:

[t,y] = ode45(odefun,tspan,y0)
[t,y] = ode45(odefun,tspan,y0,options)

Описание:

[t,y] = ode45(odefun,tspan,y0),
где tspan = [t0 tf],
интегрирует систему дифференциальных уравнений y’=f(t,y) от t0 к tf с начальными условиями y0. Каждая строка в массиве решения y соответствует значению, возвращенному в вектор-столбце t

[t,y] = ode45(odefun,tspan,y0,options)
использует настройки интегрирования, заданные options – структура опции. Option создается с использованием функции odeset. Например, используйте AbsTol и RelTol опции, чтобы задать допуски абсолютной и относительной погрешности.

Например:

```
options = odeset('RelTol',1e-8,'AbsTol',1e-10);
```

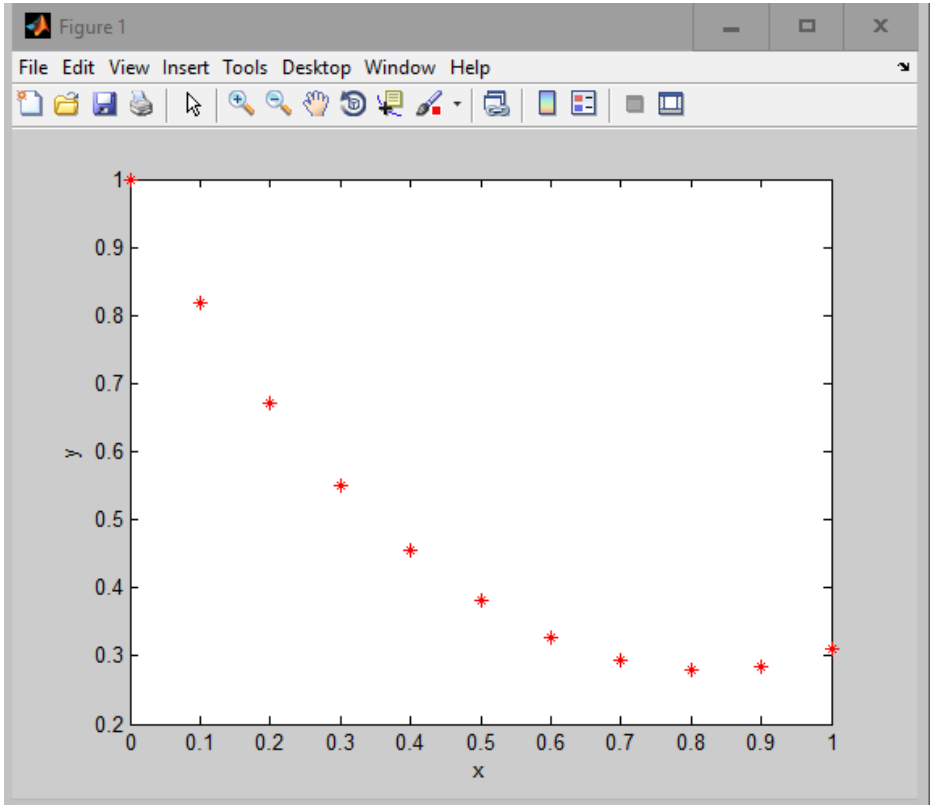
9.3.4 Пример решения обыкновенного дифференциального уравнения в Matlab

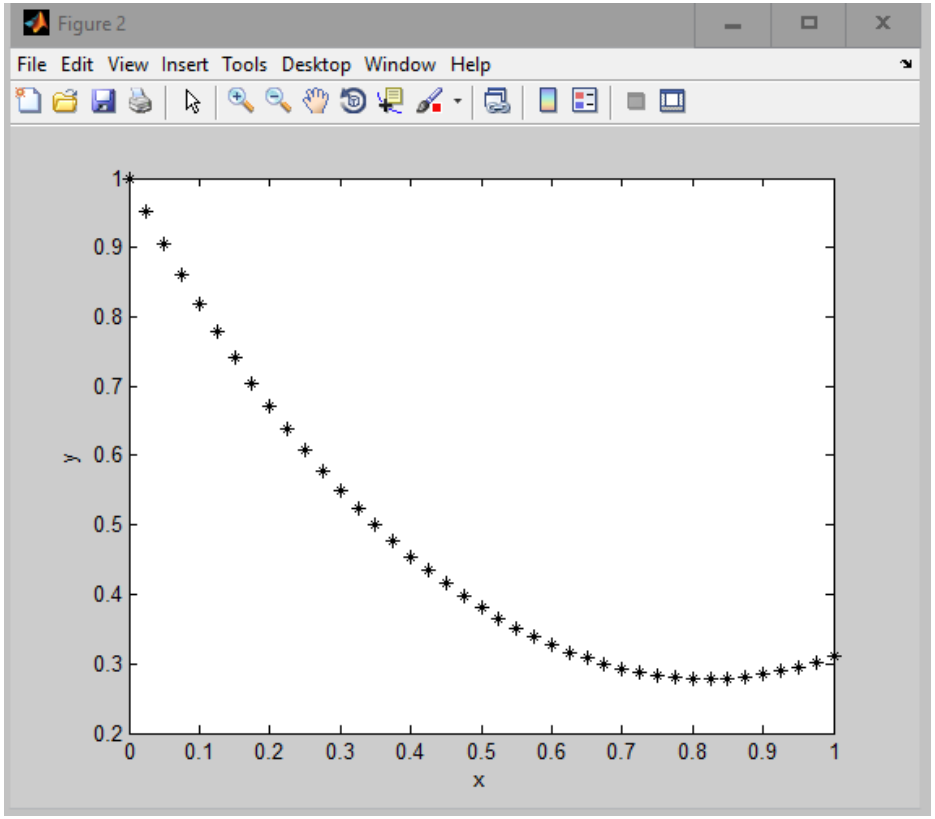
Найдем численное и аналитическое решение задачи Коши для дифференциального уравнения на интервале [0,1]:

$$\frac{dy}{dx} = x^3 - 2y \qquad (9.16)$$
$$y(0) = 1$$

```
clc;
%Определяем частное решение диф.ур. при y(0) = 0 (решение задачи Коши)
FDY = dsolve('Dy = x^3 - 2*y','y(0) = 1','x')
v = symvar(FDY); % получаем список переменых
dya = @(X) double(subs(FDY,v,X)); % создаем функцию-аналитического решения
% % Решаем на интервале [0,1] по x
x = 0:0.1:1;
y0 = 1;
n = length(x);
ya = dya(x);
plot(x,ya,'r*');
%Решаем встренным решателем ode45
tspan = [x(1) x(n)]; % решаем на интервале [0,1]
options = odeset('RelTol',1E-3);
[xx,y] = ode45(@(xx,y) xx^3 - 2*y, tspan,y0,options);
figure; plot(x,ya,'r*',xx,y,'k*');
```

9.3.4 Пример решения обыкновенного дифференциального уравнения в Matlab





9.3.5 Пример решения жесткого дифференциального уравнения в Matlab

Рассмотрим пример из литературы, относящийся к динамике. Предположим, что есть частица, с координатами ($x(t)$,$y(t)$), и требуется чтобы ее y-координата всегда оставалась равной нулю. Один из способов добиться этого — добавить слагаемое $-k \cdot y(t)$, к производной $y'(t)$, где k — большая положительная постоянная. Если k достаточно велико, то частица никогда не уйдет далеко от $y(t) = 0$, так как слагаемое $k \cdot y(t)$ всегда приведет $y(t)$ обратно к нулю. Для того, чтобы перемещать частицу как угодно вдоль оси x необходимо дифференциальное уравнение следующего вида:

$$\dot{X}(t) = \begin{cases} \dfrac{dx(t)}{dt} = -x(t) \\ \dfrac{dy(t)}{dt} = -k \cdot y(t) \end{cases} = \begin{pmatrix} -x(t) \\ -k \cdot y(t) \end{pmatrix} \qquad (9.16)$$

Предполагаем, что $y_0 \neq 0$. В этом случае частица будет сильно притягиваться к прямой $y = 0$, и менее сильно — к $x = 0$. Если решать ОДУ на достаточно продолжительном интервале времени, то частица рано или поздно попадет в точку (0,0) и останется в ней.

9.3.5 Пример решения жесткого дифференциального уравнения в Matlab

Рассмотрим пример из литературы, относящийся к динамике. Предположим, что есть частица, с координатами ($x(t)$,$y(t)$), и требуется чтобы ее y-координата всегда оставалась равной нулю. Один из способов добиться этого — добавить слагаемое $-k \cdot y(t)$, к производной $y'(t)$, где k — большая положительная постоянная. Если k достаточно велико, то частица никогда не уйдет далеко от $y(t) = 0$, так как слагаемое $k \cdot y(t)$ всегда приведет $y(t)$ обратно к нулю. Для того, чтобы перемещать частицу как угодно вдоль оси x необходимо дифференциальное уравнение следующего вида:

$$\dot{X}(t) = \begin{cases} \dfrac{dx(t)}{dt} = -x(t) \\ \dfrac{dy(t)}{dt} = -k \cdot y(t) \end{cases} = \begin{pmatrix} -x(t) \\ -k \cdot y(t) \end{pmatrix} \qquad (9.16)$$

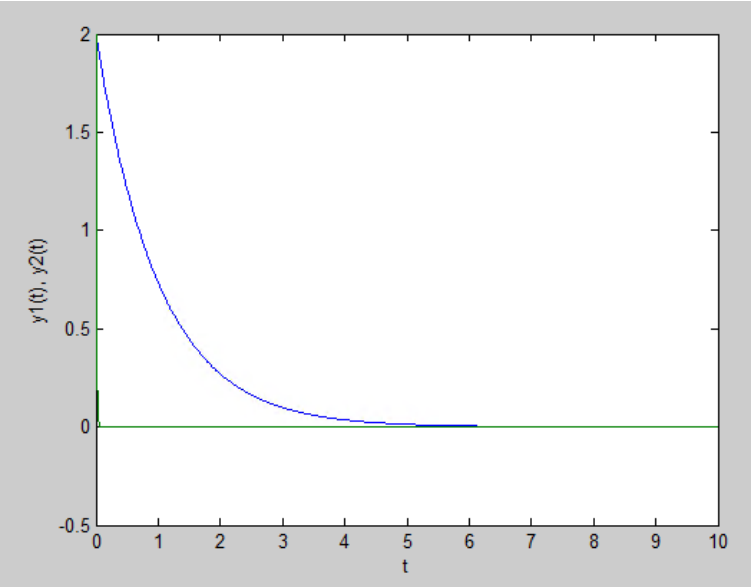
Предполагаем, что $y_0 \neq 0$. В этом случае частица будет сильно притягиваться к прямой $y = 0$, и менее сильно — к $x = 0$. Если решать ОДУ на достаточно продолжительном интервале времени, то частица рано или поздно попадет в точку (0,0) и останется в ней.

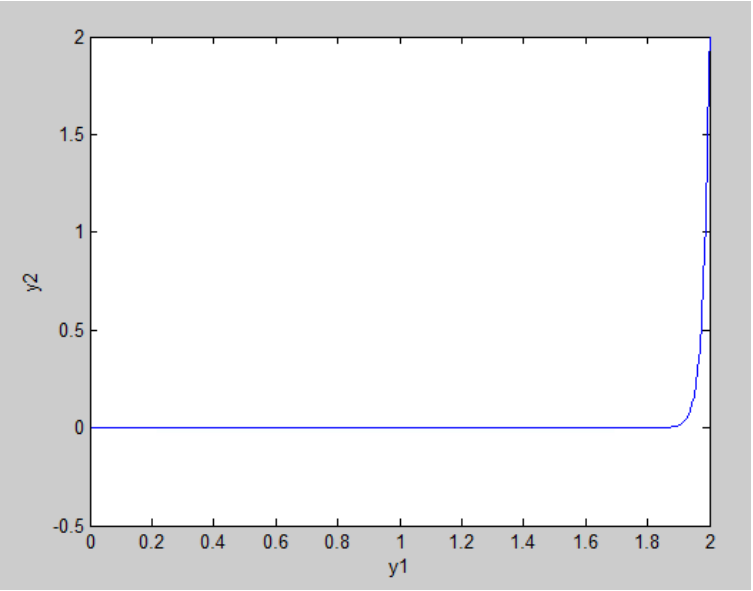
9.3.5 Пример решения жесткого дифференциального уравнения в Matlab

f2.m

```
function dydt = f2(t,y)
k = 100;
dydt = zeros(2,1);
dydt(1) = -y(1);
dydt(2) = -k*y(2);
```

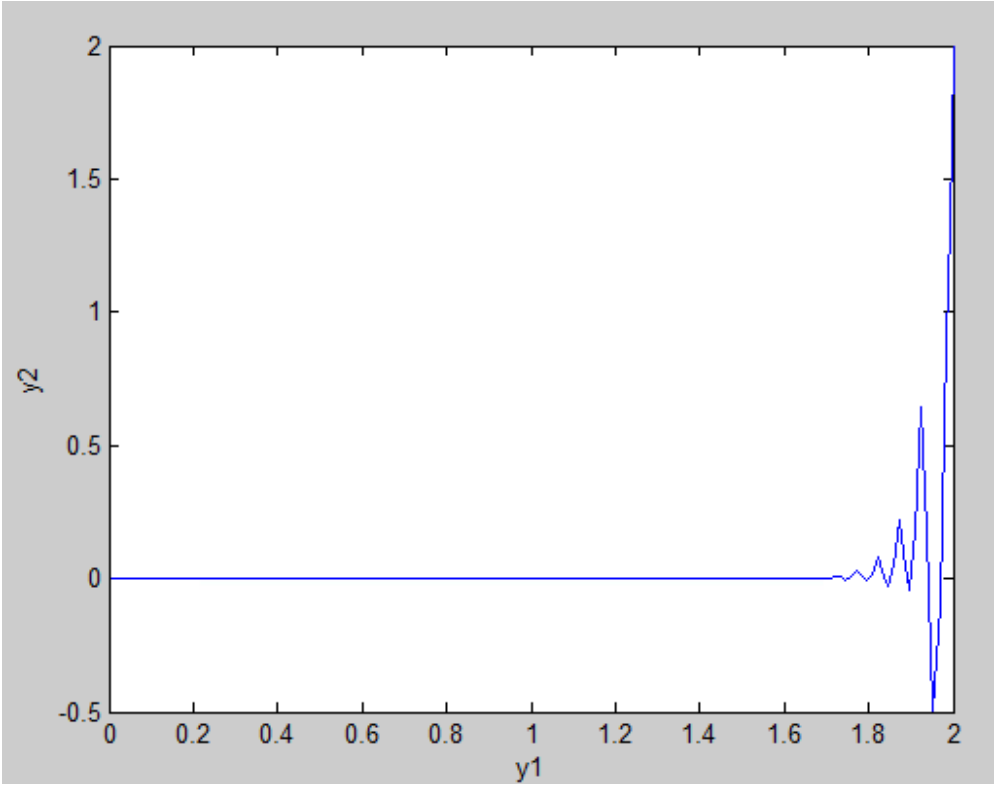
```
clc;
y0 = [2 2];
tspan = [0 10];
options = odeset('RelTol',1E-2);
[t,y] = ode45(@f2,tspan,y0,options);
n = length(y);
XX = [];
YY = [];
for i=1:n
    XX(i) = y(i,1);
    YY(i) = y(i,2);
end
plot(t,y);
figure;plot(XX,YY);
```





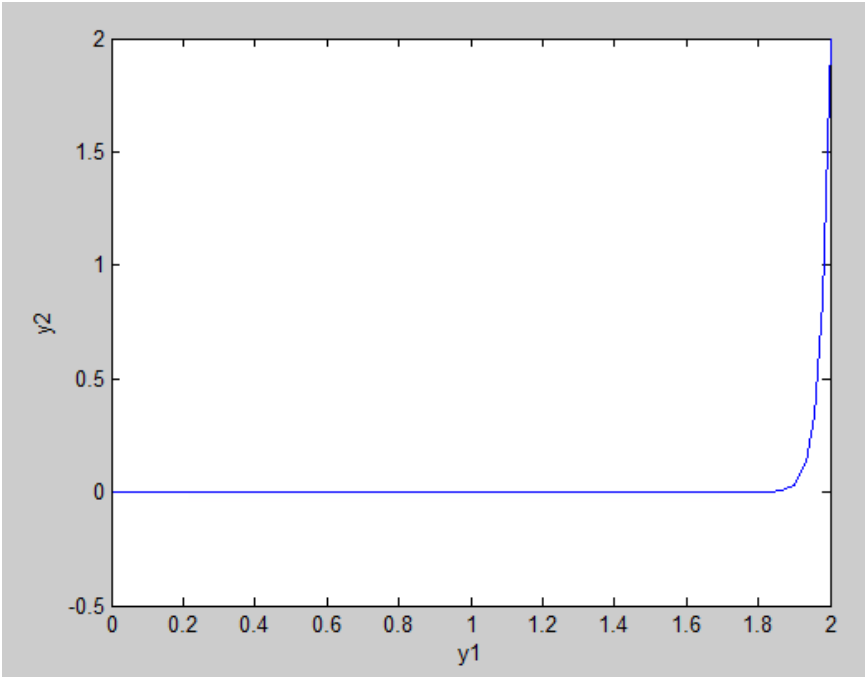
9.3.5 Пример решения жесткого дифференциального уравнения в Matlab (увеличиваем допустимую ошибку до 1%)

```
1 -    clc;
2 -    y0 = [2 2];
3 -    tspan = [0 10];
4 -    options = odeset('RelTol',1);
5 -    [t,y] = ode45(@f2,tspan,y0,options);
6 -    n = length(y);
7 -    XX = [];
8 -    YY = [];
9 -    for i=1:n
10 -       XX(i) = y(i,1);
11 -       YY(i) = y(i,2);
12 -   end
13 -   plot(t,y);
14 -   figure;plot(XX,YY);
```



9.3.5 Пример решения жесткого дифференциального уравнения в Matlab (увеличиваем допустимую ошибку до 1%, ode15s)

```
1  clc;
2  y0 = [2 2];
3  tspan = [0 10];
4  options = odeset('RelTol',1);
5  [t,y] = ode15s(@f2,tspan,y0,options);
6  n = length(y);
7  XX = [];
8  YY = [];
9  for i=1:n
10     XX(i) = y(i,1);
11     YY(i) = y(i,2);
12 end
13 plot(t,y);
14 figure;plot(XX,YY);
```



9.4 Моделирование динамических систем, описываемых системами ОДУ в Simulink

Simulink® является средой блок-схемы для многодоменной симуляции и Модельно-ориентированного проектирования. Поддерживает разработку системы, симуляцию, автоматическую генерацию кода, и непрерывный тест и верификацию встраиваемых систем. Simulink предоставляет графический редактор, настраиваемые библиотеки блоков и решатели для моделирования и симуляции динамических систем. Также обеспечивает интеграцию с MATLAB®, позволив вам включить алгоритмы Matlab в модели и экспортировать результаты симуляции в MATLAB для последующего анализа.

1. Самостоятельно ознакомиться с пакетом расширения Simulink https://docs.exponenta.ru/simulink/getting-started-with-simulink.html?s_tid=CRUX_lftnav

2. Выполним моделирование системы (9.16) в Simulink

9.4 Моделирование динамических систем, описываемых системами ОДУ в Simulink