Дифференциальные уравнения

Обыкновенные дифференциальные уравнения

$$\frac{dy}{dt} = f(t, y),$$

Задача Коши

$$y(t_0) = y_0$$

Решение дифференциальных уравнений

Для решения дифференциальных уравнений предназначены специальные функции MATLAB, в вычислительной математике их называют ***солверы***.

MATLAB имеет достаточно большой набор ***солверов***, основанных на различных численных методах.

Решение задачи Коши

Для решения задачи Коши в MATLAB существует семь ***солверов***: **ode45**, **ode23**, **ode113**, **ode15s**, **ode23s**, **ode23t**, **ode23tb**. Методика их использования одинакова, включая способы задания входных и выходных аргументов.

В **Octave** применяются четыре совместимых с MATLAB ***солверов*****: ode45**, **ode23, ode15s, ode15i.**

В достаточно общем случае вызов ***солвера*** для решения задачи Коши производится следующим образом (здесь под ***solver*** понимается один из перечисленных выше ***солверов***):

[T, Y]=solver(odefun,interval,y0,options)

Дифференциальное уравнение должно быть приведено к виду:

y’=f(t,y), y(0)=y0 – система дифф. уравнений; y,y’ –векторы-столбцы, t – скаляр.

[T, Y] = solver(odefun,interval,y0,options),

где ***odefun*** —указатель на вектор функцию- столбец правой части системы уравнений, ***interval*** — вектор-строка из двух чисел, задающая промежуток для решения уравнения, ***y0*** — заданный вектор-столбец начальных значений искомой вектор-функции, ***options***— структура для управления параметрами и ходом вычислительного процесса.

Солвер возвращает массив ***T*** с координатами узлов сетки, в которых найдено решение, и матрицу решений ***Y***, каждый столбец которой является значением компоненты вектор-функции решения в узлах сетки.

Задача Коши для дифференциального уравнения состоит в нахождении функции, удовлетворяющей дифференциальному уравнению произвольного порядка:

$$y^{(n)}=f(t,y^{(n-1)},y^{(n-2)},\ldots,y),$$

с начальными условиями:

$$y(0)=y_0,y^{(1)}(0)=y_1,\ldots,y^{(n-1)}(0)=y_{n-1}$$

Схема решения таких задач в MATLAB состоит из следующих этапов.

1. Приведение дифференциального уравнения к системе дифференциальных уравнений первого порядка.
2. Написание специальной функции для системы уравнений.
3. Вызов подходящего ***солвера***.
4. Визуализация результата.

Разберем решение дифференциальных уравнений на примере задачи о колебаниях материальной точки под воздействием внешней силы в среде, оказывающей сопротивление колебаниям. Перемещение точки в среде описывается уравнением второго порядка

$$y'' + 2y' + 10y = \sin t.$$

Предположим, что координата точки в начальный момент времени равнялась единице, а скорость — нулю. Тогда соответствующие начальные условия имеют вид: $y(0)=1, y'(0)=0$

Исходную задачу надо привести к системе дифференциальных уравнений. Для этого вводят столько вспомогательных функций, каков порядок уравнения. В данном случае необходимы две вспомогательные функции y_1 и y_2 , определяемые формулами:

$y_1=y,\ y_2=y'.$

Система дифференциальных уравнений с начальными условиями, требуемая для дальнейшей работы, такова:

$$\begin{cases} y_1' = y_2 & y_1(0)=1 \\ y_2' = -2y_2 - 10y_1 + \sin(t) & y_2(0)=0 \end{cases}$$

Второй этап состоит в написании функции для системы дифференциальных уравнений. Функция должна иметь два входных аргумента: переменную t, по которой производится дифференцирование, и вектор, размер которого равен числу неизвестных функций системы. Число и порядок аргументов фиксированы, даже если *t* явно не входит в систему. Выходным аргументом функции является вектор правой части системы.

При выборе *солвера* для решения задачи необходимо учитывать свойства системы дифференциальных уравнений, иначе можно получить неточный
результат или затратить слишком много времени на решение.

```
% Решение системы дифференциальных уравнений
% разными солверами
% Анонимная функция вычисления правой части
f=@(t,y)[y(2);-2*y(2)-10*y(1)+sin(t)];
% Начальные условия
y0 = [1; 0];
% Вызов солвера
[T, Y] = ode45(f, [0 15], y0); %солверы можно менять
%Построение графика с пояснениями
plot(T, Y(:, 1), 'r.-',T, Y(:, 2), 'k.:' )
title('Solve {\ity} \prime\prime+2{\ity} \prime+10{\ity} =
sin{\itt}' )
xlabel('\itt' )
ylabel( '{\ity}, {\ity} \prime ' )
legend( 'Y', 'dy/dt', 4)
grid on
%здесь применено
```

Здесь применено кодирование символов как в LaTex:

\bf Bold font

\it Italic font

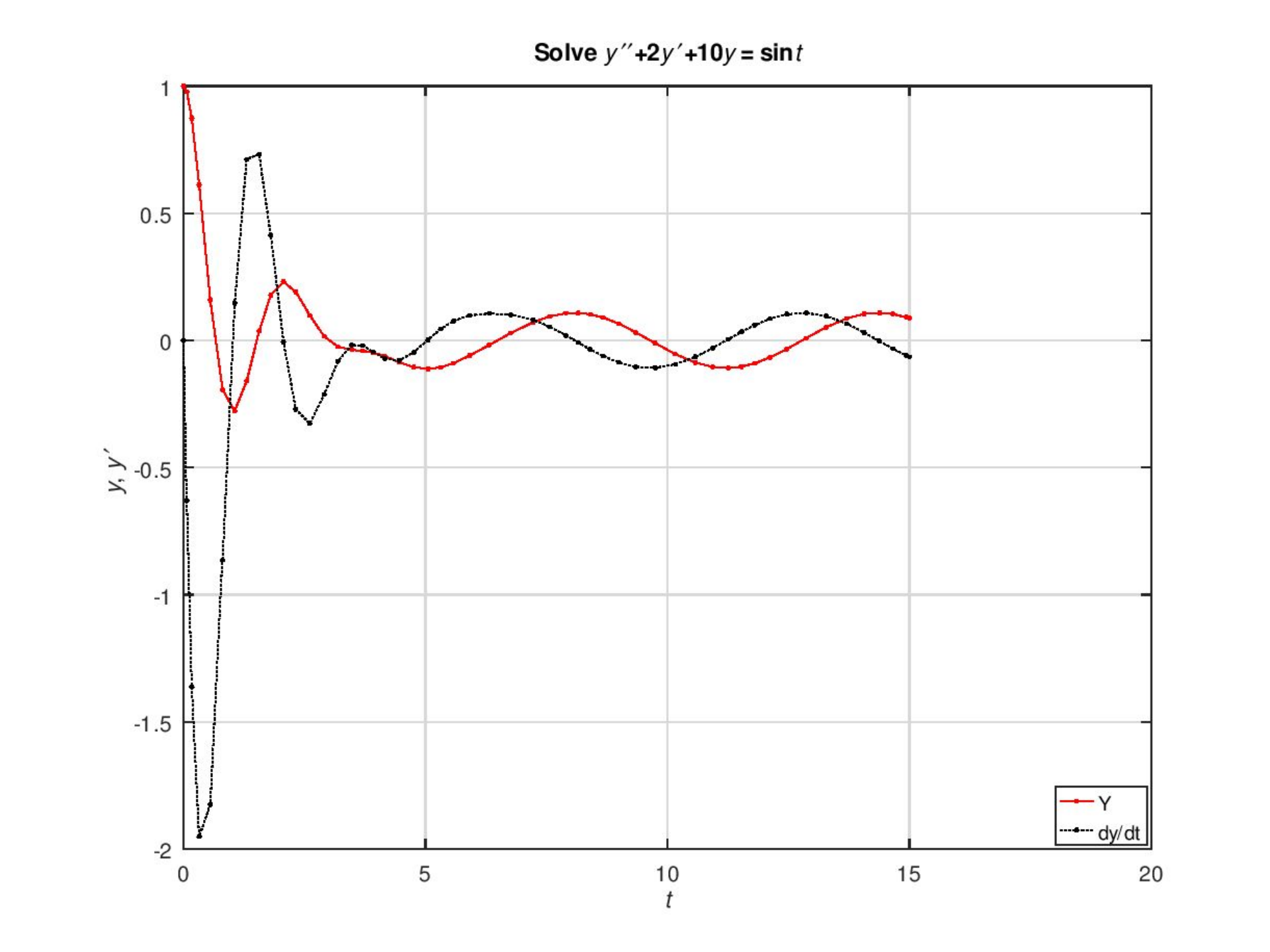
\rm Normal font

Символы должны быть заключены в фигурные скобки: {\bfPrimer}

Данное кодирование допускается только при выводе графиков в функциях title, legend,label

Детальное описание дано в п. 15.2.8 Документации

Можно использовать греческие буквы, математические символы и т.д.
Очень хороший способ документирования графиков.


Solve y′′+2y′+10y = sin t
1
0.5
0
-0.5
-1
-1.5
-2
y, y′
0
5
10
15
20
t
Y
dy/dt

Задание: Привести следующие дифференциальные уравнения к системе дифференциальных уравнений первого порядка

y’’’’=-t^2*y’’’-y*sin(t)-y’

y(0)=0,y’(0)=1,y’’(0)=0,y’’’(0)=0.

y’’=2(1-y^2)y-y’, y(0)=0, y’(0)=1.

Решить эти уравнения и построить графики решений. Результат выслать мне.

```
function solvdem
% Файл-функция для решения дифференциального уравнения
% формирование вектора начальных условий
y0 = [1; 0];
% вызов солвера от функции oscil, начального и конечного
% момента времени и вектора начальных условий
[T, Y] = ode113(@oscil, [0 15], y0);
% вывод графика решения исходного дифференциального уравнения
% (маркеры - точки, линия - сплошная)
plot(T, Y(:, 1), 'r.-')
% вывод графика производной от решения исходного
% дифференциального уравнения (маркеры - точки, линия - пунктир)
hold on
plot(T, Y(:, 2), 'k.:‘ )
% вывод пояснений на график
title('Решение {\ity} \prime\prime+2{\ity} \prime+10{\ity} = sin{\itt}‘ )
xlabel('\itt‘ )
ylabel( '{\ity}, {\ity} \prime ‘ )
legend( 'координата', 'скорость', 4)
grid on
hold off
% подфункция вычисления правых частей уравнений
function F = oscil(t, y)
F = [y(2); -2*y(2) - 10*y(1) + sin(t)];
```

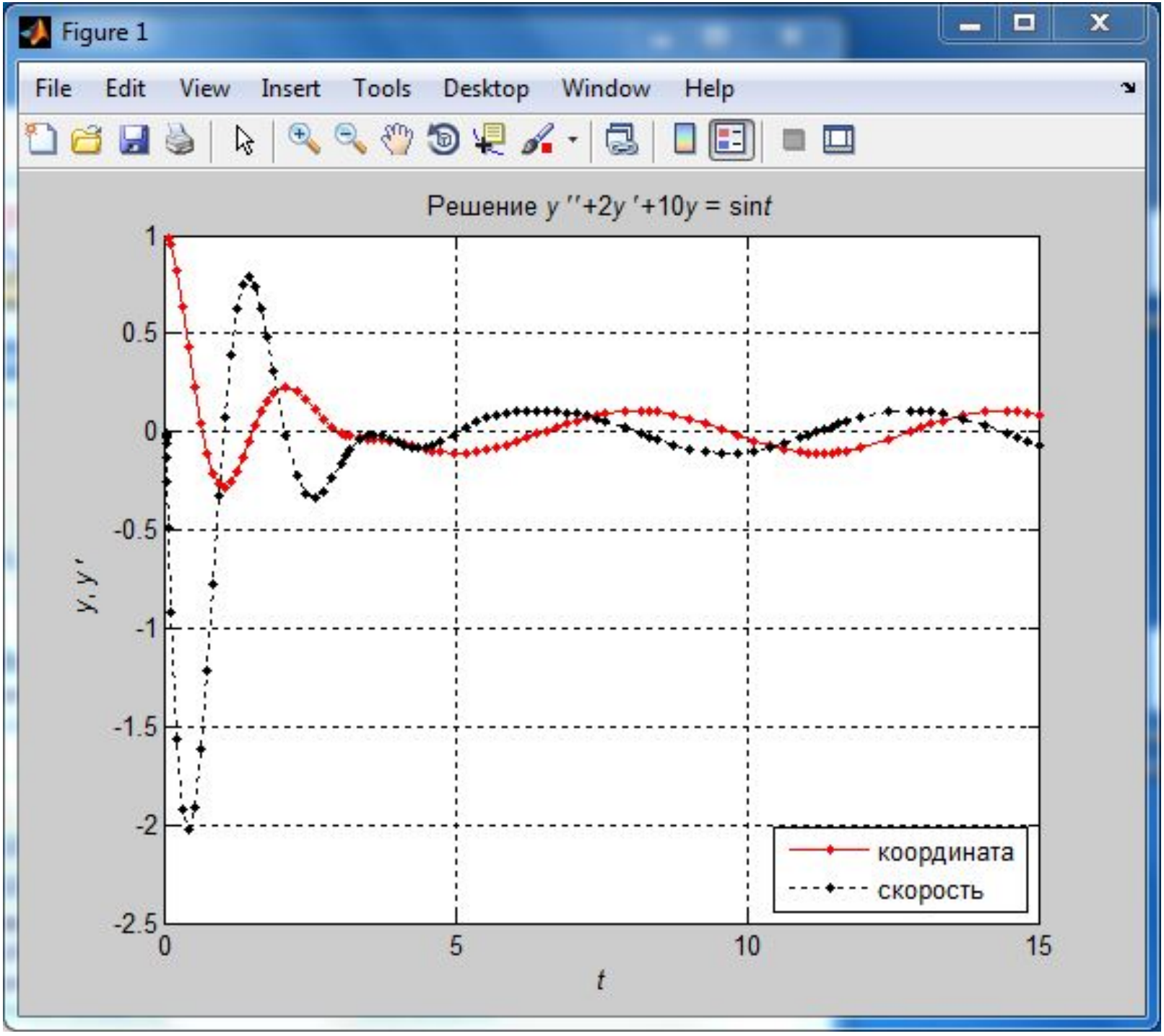


Figure 1
File Edit View Insert Tools Desktop Window Help
Решение y ''+2y '+10y = sint
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
y, y '
0
5
10
15
t
координата
скорость

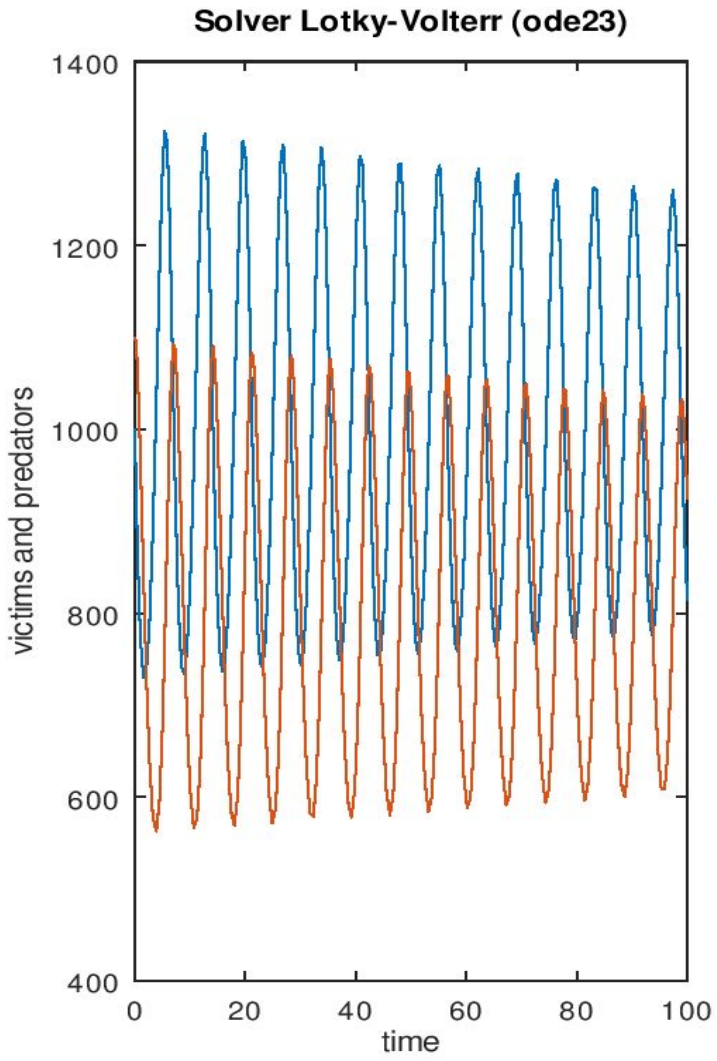
Решение уравнений Лотки—Вольтерры разными солверами

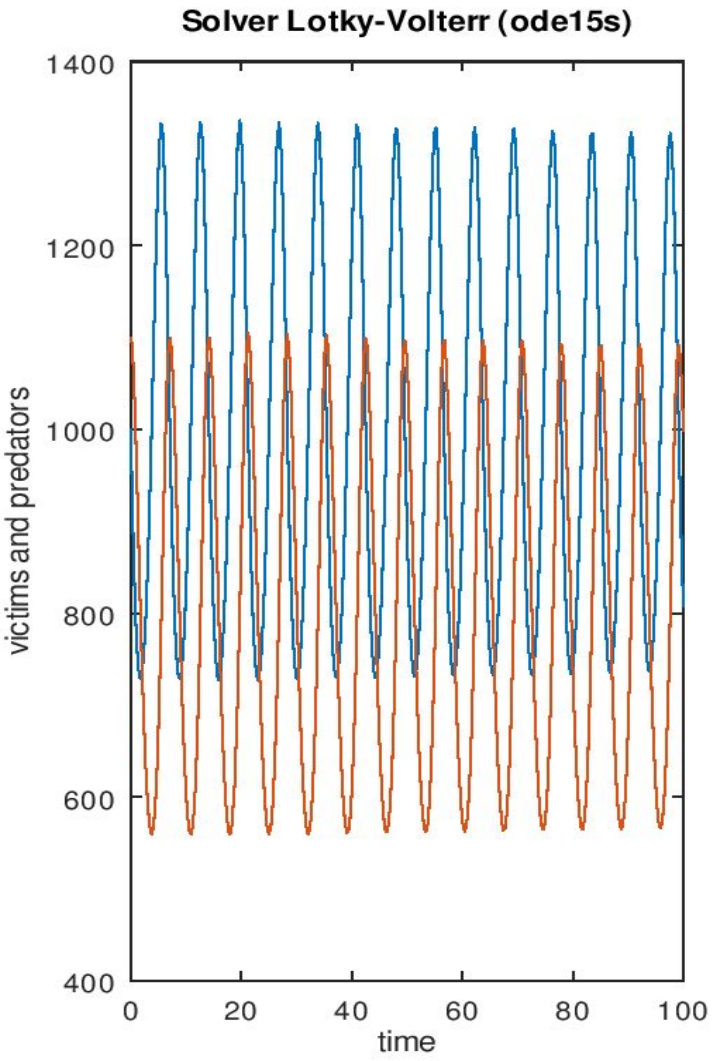
В качестве объекта исследования возьмем модель Лотки—Вольтерры борьбы за существование. Обозначим: $y_1(t)$ — число жертв, $y_2(t)$ — число хищников. Число хищников и жертв в течение времени t изменяется по закону

$$\begin{cases} y_1' = Py_1 - py_1y_2 \\ y_2' = -Ry_2 + ry_1y_2 \end{cases}$$

где P— константа увеличения числа жертв в отсутствие хищников, R— константа уменьшения числа хищников в отсутствие жертв. Вероятность поедания хищником жертвы пропорциональна их числу y_1y_2 При этом слагаемое py_1y_2 соответствует вымиранию жертв, а ry_1y_2 — появлению хищников. Возьмем для примера $P = 0.8$, $R = 1$, $p = r = 0.001$, положим, что в начальный момент времени было 1000 жертв и 1100 хищников.

```
function comparesolvers
% Сравнение солверов
Y0 = [1000; 1100];
% вызов солвера ode23
[T, Y] = ode23(@LotVol, [0 100], Y0) ;
% вывод графика решения исходного дифференциального уравнения
subplot(1, 2, 1)
plot(T,Y(:, 1),T, Y(:, 2))
% вывод пояснений на график
title('Solver Lotky—Volterr (ode23)')
xlabel('time')
ylabel( 'victims and predators' )
%вызов солвера ode15s
[T, Y] = ode15s(@LotVol, [0 100], Y0) ;
% вывод графика решения исходного дифференциального уравнения
subplot(1, 2, 2)
plot(T,Y(:, 1),T, Y(:, 2))
% вывод пояснений на график
title( 'Solver Lotky—Volterr (ode15s)' )
xlabel( 'time' )
ylabel( 'victims and predators' )
% подфункция вычисления правых частей уравнений
function F = LotVol(t, Y)
F = [0.8*Y(1) - 0.001*Y(1)*Y(2); -1.0*Y(2) + 0.001*Y(1)*Y(2)];
```





Солвер ode15s обладает большей точностью в данном случае.

```
function comparesolvers
% Сравнение солверов
Y0 = [1000; 1100];
% вызов солвера odell3
[T, Y] = ode113(@LotVol, [0 100], Y0) ;
% вывод графика решения исходного дифференциального уравнения
subplot(1, 2, 1)
plot(T,Y(:, 1),T, Y(:, 2))
% вывод пояснений на график
title('Решение уравнений Лотки—Вольтерры (odell3)')
xlabel('время')
ylabel( 'жертвы и хищники‘ )
%вызов солвера ode23s
[T, Y] = ode23s(@LotVol, [0 100], Y0) ;
% вывод графика решения исходного дифференциального уравнения
subplot(1, 2, 2)
plot(T,Y(:, 1),T, Y(:, 2))
% вывод пояснений на график
title( 'Решение уравнений Лотки—Вольтерры (ode23s)‘ )
xlabel( 'время‘ )
ylabel( 'жертвы и хищники‘ )
% подфункция вычисления правых частей уравнений
function F = LotVol(t, Y)
F = [0.8*Y(1) - 0.001*Y(1)*Y(2); -1.0*Y(2) + 0.001*Y(1)*Y(2)];
```

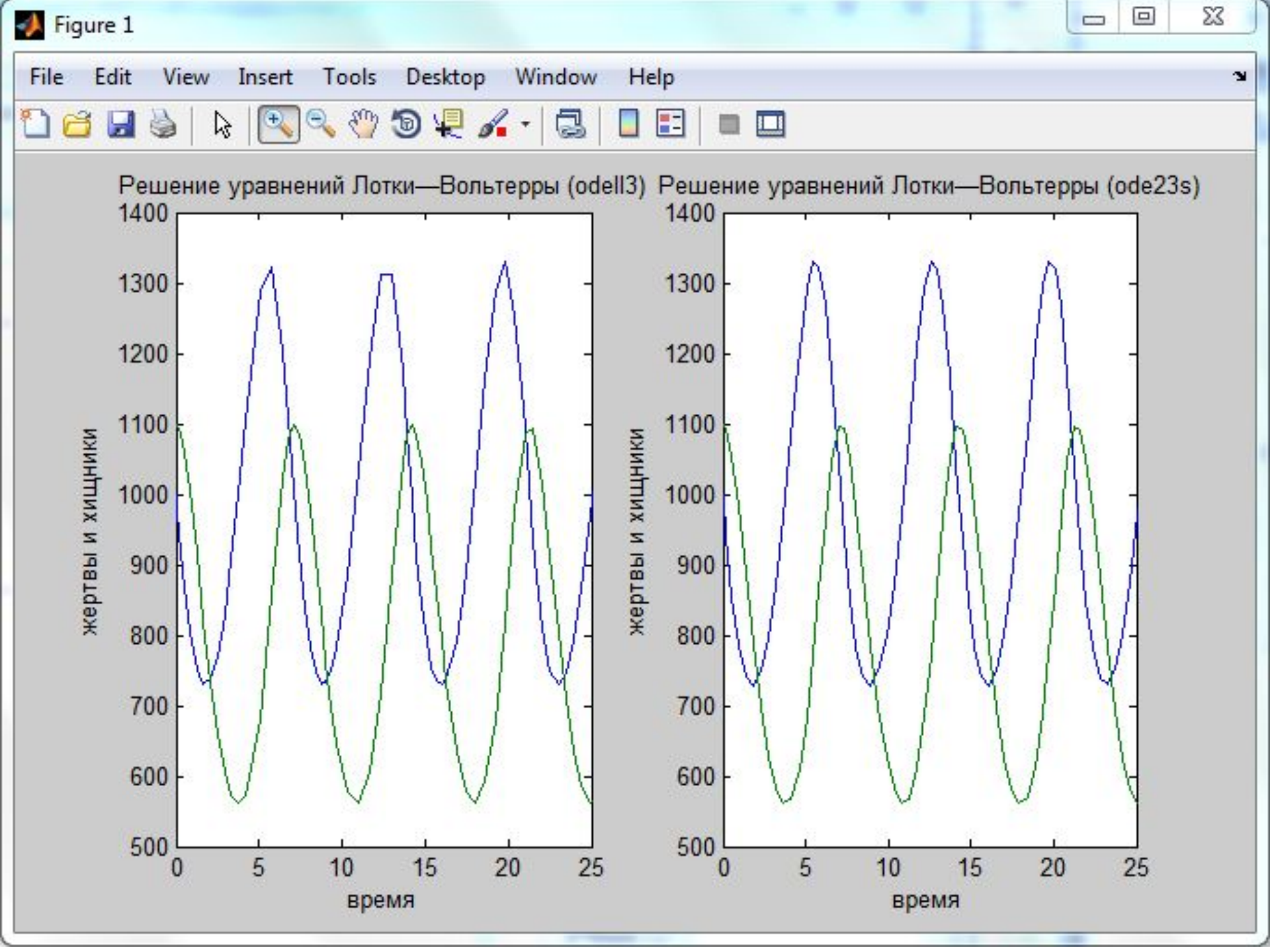


Figure 1
File Edit View Insert Tools Desktop Window Help
Решение уравнений Лотки—Вольтерры (odell3)
Решение уравнений Лотки—Вольтерры (ode23s)
жертвы и хищники
время
1400
1300
1200
1100
1000
900
800
700
600
500
0 5 10 15 20 25

Управление процессом решения

Эффективное решение дифференциальных уравнений невозможно без понимания основных вопросов, связанных с численными методами. ***Солверы*** MATLAB не являются "черными ящиками". Пользователю необходимо выбрать подходящий ***солвер***, в зависимости от свойств решаемой задачи, и произвести необходимые установки, обеспечивающие получение приближенного решения с требуемыми свойствами, например, с заданной точностью. ***Солверы*** допускают указание параметров для контроля и управления вычислительным процессом. Способ задания параметров ***солверов*** **ode45, ode23, ode113 , ode15s , ode23s, ode23t, ode23tb** аналогичен тому, который применяли при нахождении корней функции или локальных минимумов. Значения параметров записываются в управляющую структуру, которая создается функцией **odeset**.

odestruct = **odeset** *()*
odestruct = **odeset** *("field1", value1, "field2", value2, …)*
odestruct = **odeset** *(oldstruct, "field1", value1, "field2", value2, …)*
odestruct = **odeset** *(oldstruct, newstruct)*
odeset *()*
Создает или модифицирует структуру выбора ОДУ

Параметры odeset

Группа	**Имя параметра (вид контроля)**
Контроль точности вычислений	RelTol, AbsTol, NormControl
Шаг интегрирования	InitialStep , MaxStep
Выходные данные	OutputFcn,OutputSel, Refine,Stats

Структура с опциями солверов модифицируется так же, как и в случае с ***optimset*** при решении уравнений и минимизации функций:

options = odeset(options, вид_контроля, значение,...)

Вызов ***odeset*** без входных аргументов позволяет посмотреть в командном окне имена всех свойств и их возможные значения, причем в фигурных скобках указаны значения, используемые солверами по умолчанию.

Функция ***odeget*** предназначена для извлечения значения свойства из структуры

значение = odeget(options, вид_контроля)

```
>> odeset()
List of the most common ODE solver options.
Default values are in square brackets.

            AbsTol:  scalar or vector, >0, [1e-6]
               BDF:  binary, {["off"], "on"}
            Events:  function_handle, []
      InitialSlope:  vector, []
       InitialStep:  scalar, >0, []
          Jacobian:  matrix or function_handle, []
         JConstant:  binary, {["off"], "on"}
          JPattern:  sparse matrix, []
              Mass:  matrix or function_handle, []
      MassSingular:  switch, {["maybe"], "no", "yes"}
          MaxOrder:  switch, {[5], 1, 2, 3, 4, }
           MaxStep:  scalar, >0, []
  MStateDependence:  switch, {["weak"], "none", "strong"}
        MvPattern:  sparse matrix, []
       NonNegative:  vector of integers, []
       NormControl:  binary, {["off"], "on"}
         OutputFcn:  function_handle, []
         OutputSel:  scalar or vector, []
            Refine:  scalar, integer, >0, []
            RelTol:  scalar, >0, [1e-3]
             Stats:  binary, {["off"], "on"}
      Vectorized:  binary, {["off"], "on"}
```

Задание точности вычислений

Точность вычислений оказывает существенное влияние на качество приближенного решения.
Допускается два способа контроля точности в зависимости от значения параметра ***NormControl***:
1. По локальной погрешности e_i *i*-ой (y_i) компоненты вектора решений, если параметр ***NormControl*** имеет значение ***off*** (по умолчанию).
2. По евклидовой норме погрешности, для ***NormControl***, установленного в ***on*.**
В первом случае точность считается достигнутой при выполнении условий $e_i(t_k) < \max\{RelTol\ |y_i(t_k)|\ ,\ AbsTol\ (i)\}$ для всех компонент вектора решений на каждом шаге по времени t_k . Во втором случае принимается во внимание суммарная характеристика для всех компонент решения — евклидова норма вектора || || (вычисляемая встроенной функцией ***norm***) — и точность считается достигнутой, если на каждом шаге по времени t_k выполняется неравенство $\|e\| < \max\{RelTol|y_i(t_k)|, AbsTol\}$. В случае покомпонентной оценки параметр ***AbsTol*** (абсолютная точность) может быть числом. Для контроля абсолютной точности каждой компоненты следует указать вектор значений. ***RelTol*** определяет число верных значащих цифр во всех компонентах решения.

Шаг интегрирования солвера определяется двумя свойствами:

• ***Maxstep*** — задает максимальный шаг (по умолчанию десятая часть промежутка интегрирования);

• ***InitialStep*** — задает начальный шаг, и если он не определен, то выбирается солвером с учетом начальных значений для вектора решений. При нулевых значениях шаг может оказаться слишком большим.

Убедимся в том, что заданных по умолчанию значений, в частности, относительной погрешности 10^{-3}, не всегда достаточно для получения хорошего приближения.

Решим систему дифференциальных уравнений:

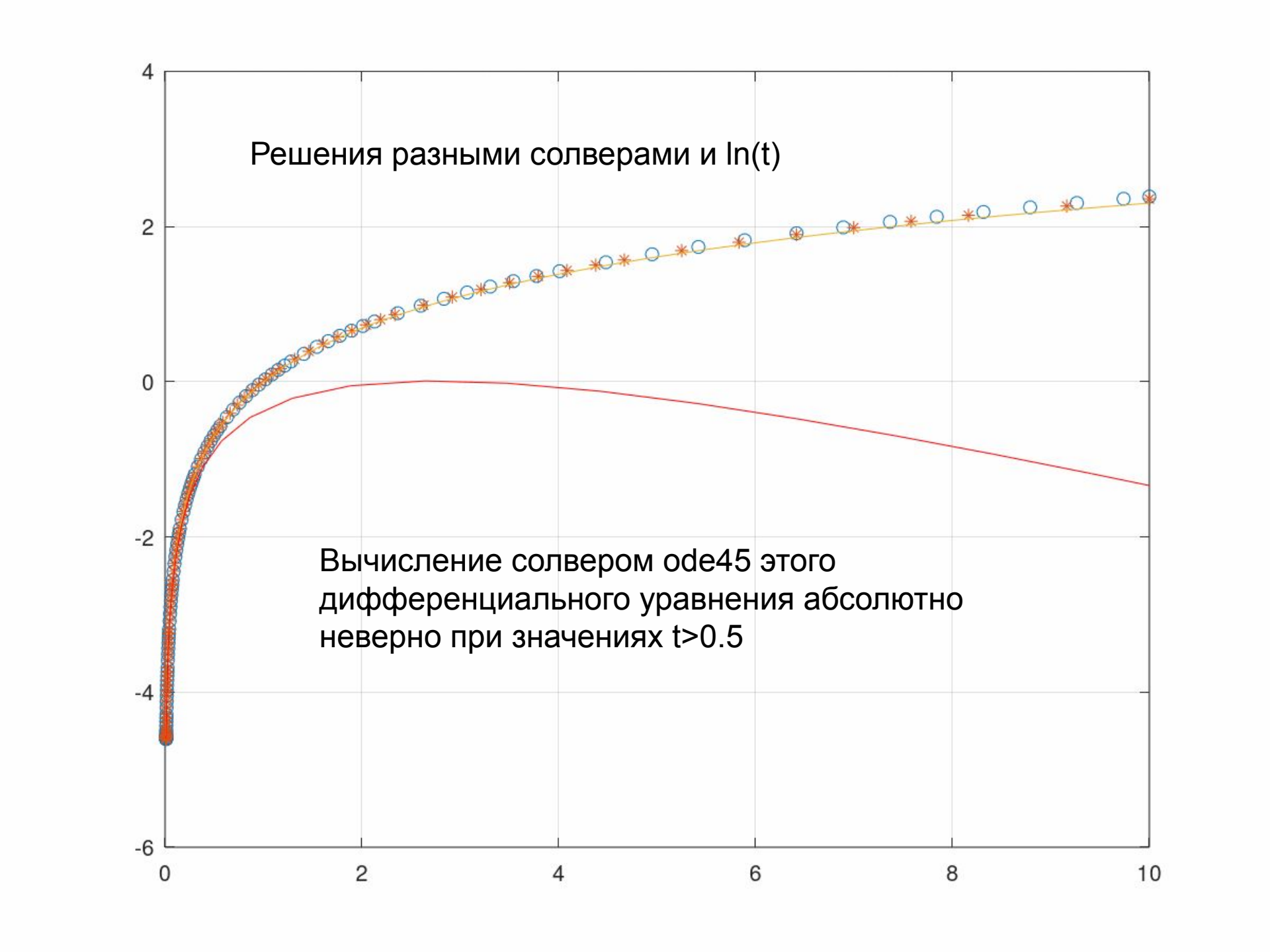
$$\begin{cases} y_1' = y_2 \\ y_2' = -1/t^2 \end{cases}$$

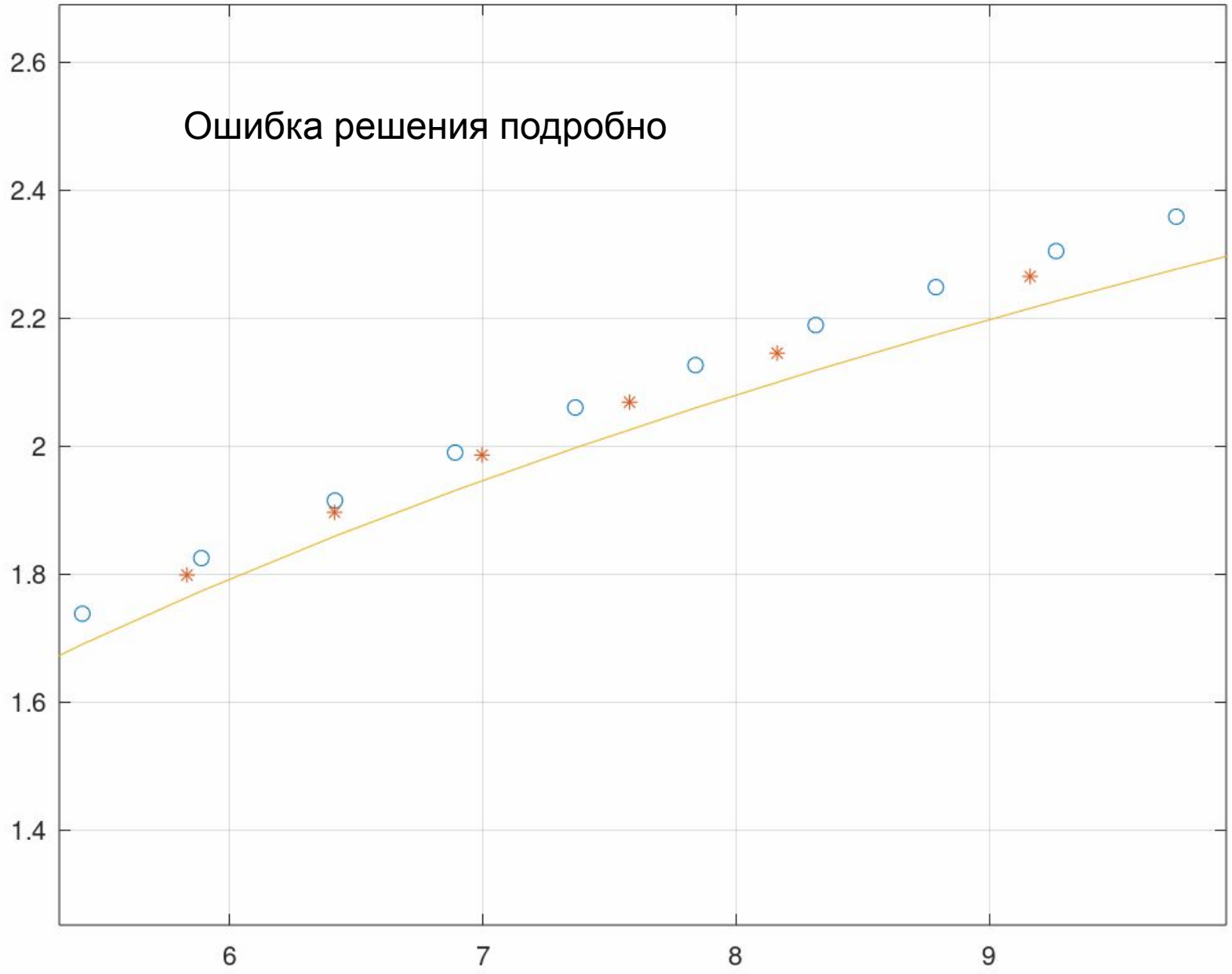
на отрезке ***[a, 10]*** при начальных условиях $y_1(a) = ln(a)$, $y_2(a) = 1/a$, взяв a = 0.01.

Точное решение: y_1=ln(t), y_2=1/t.

Написать программу решения этого уравнения с абсолютной погрешностью 1e-3.

```
% Решение дифференциального уравнения y''=-1/t^2
a=0.01;
f=@(t,y)[y(2);-1/t.^2];        % Правая часть системы
y0=[log(a);1/a];               % Вектор-столбец начальных условий
options=odeset('AbsTol',1e-6); % Управляющий параметр. Задаем абсолютную погрешность
[T,Y]=ode15s(f,[a,10],y0,options); % Вызываем солвер ode15s
[T2,Y2]=ode45(f,[a,10],y0,options); % Вызываем солвер ode45
options=odeset('AbsTol',1e-3); % Задаем абсолютную погрешность
[T1,Y1]=ode15s(f,[a,10],y0,options); % Вызываем солвер
plot(T,Y(:,1),'o',T1,Y1(:,1),'*',T,log(T),T2,Y2(:,1),'r');grid on
```


Решения разными солверами и ln(t)
Вычисление солвером ode45 этого дифференциального уравнения абсолютно неверно при значениях t>0.5
4
2
0
-2
-4
-6
0
2
4
6
8
10


Ошибка решения подробно
2.6
2.4
2.2
2
1.8
1.6
1.4
6
7
8
9

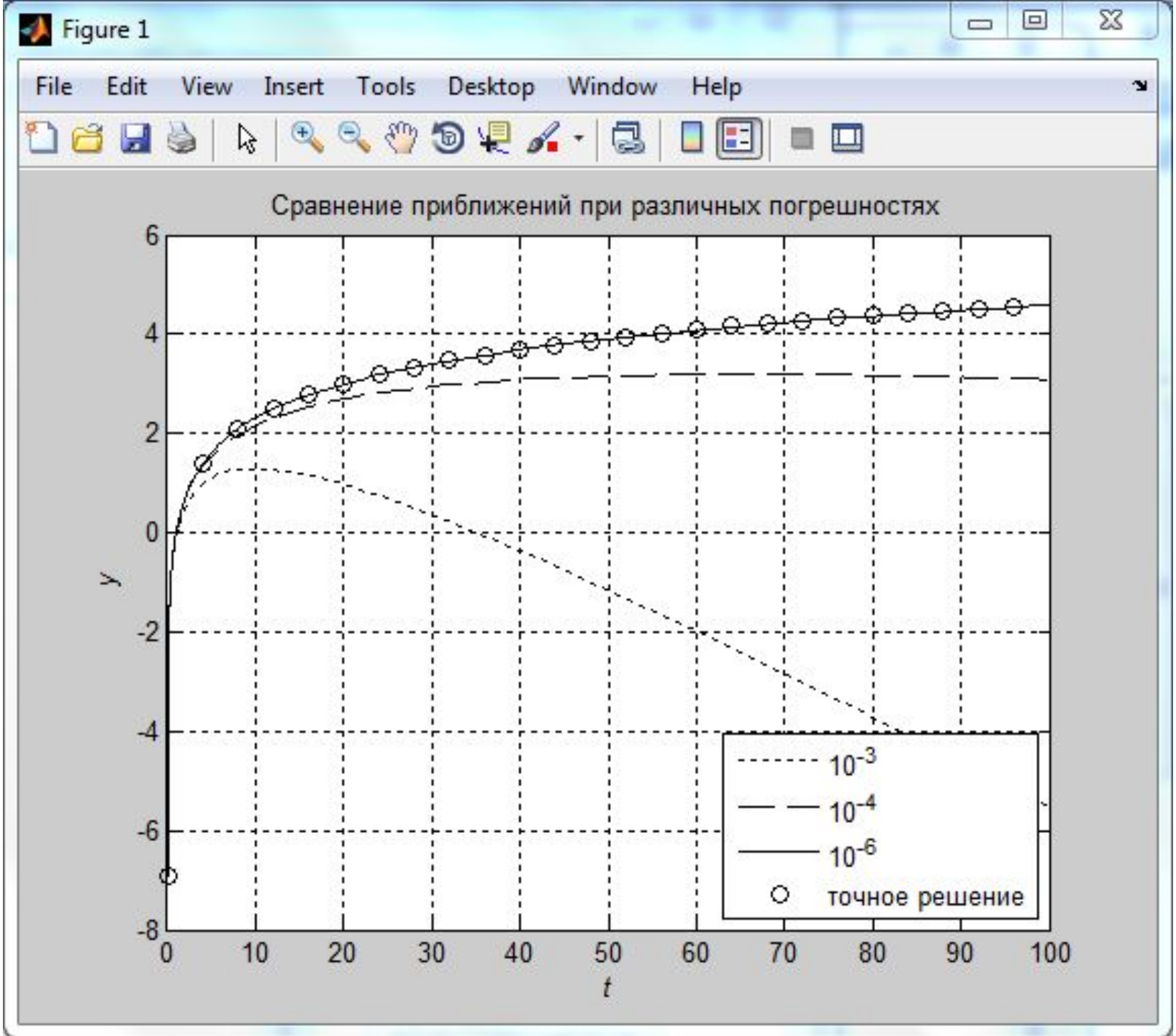


Figure 1
File Edit View Insert Tools Desktop Window Help
Сравнение приближений при различных погрешностях
6
4
2
0
-2
-4
-6
-8
y
0 10 20 30 40 50 60 70 80 90 100
t
10^-3
10^-4
10^-6
точное решение

Управление выводом результатов

Примеры предыдущих разделов предполагали вызов солверов с двумя выходными аргументами — массивами. В них возвращался вектор значений независимой переменной и матрица со значениями компонент решения в соответствующих точках. Полученные массивы мы использовали для визуализации и анализа результата. Вызов солвера без выходных аргументов приводит к появлению графического окна, в котором отображается процесс численного интегрирования дифференциального уравнения и выводятся все компоненты вектора решения.

Возможности вывода результата, предоставляемые солверами MATLAB, не исчерпываются только таким способом визуализации решения. Пользователь может выбрать альтернативное графическое представление результата, более того, допускается контролировать процесс численного интегрирования и отображать его на графике по своему усмотрению. Для этого следует воспользоваться одним из видов контроля управляющей структуры — ***outputFcn.***
Его значение должно быть указателем на функцию (или ее именем), производящую требуемые операции.

Имеется несколько стандартных функций:

- ***odeplot*** — построение графиков компонент решения;
- ***odephas2*** — построение графиков компонент решения в фазовых координатах для двумерного процесса;(MatLab)
- ***odephas3***— построение графиков компонент решения в фазовых координатах для трехмерного процесса; (MatLab)
- ***odeprint*** — печать решения. (MatLab)

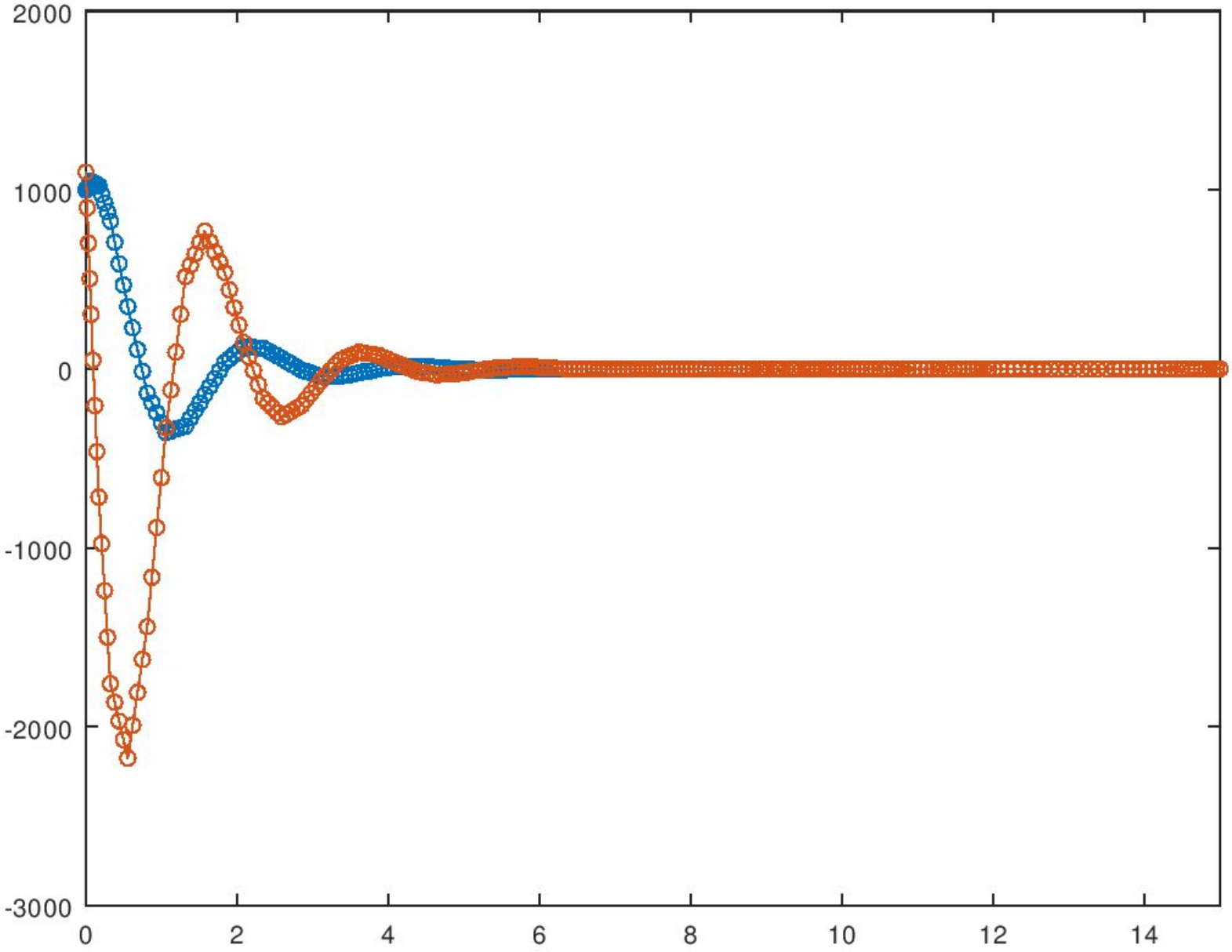
Выполните эти операторы для уравнения в начале лекции:

```
f=@(t,y)[y(2);-2*y(2)-10*y(1)+sin(t)];

options = odeset('OutputFcn', @odeplot);
[T,Y] = ode45(f, [0 15], y0, options);
```

Можно так:

```
ode45(f, [0 15], y0);
```


2000
1000
0
-1000
-2000
-3000
0
2
4
6
8
10
12
14

Дифференциальные уравнения с параметрами

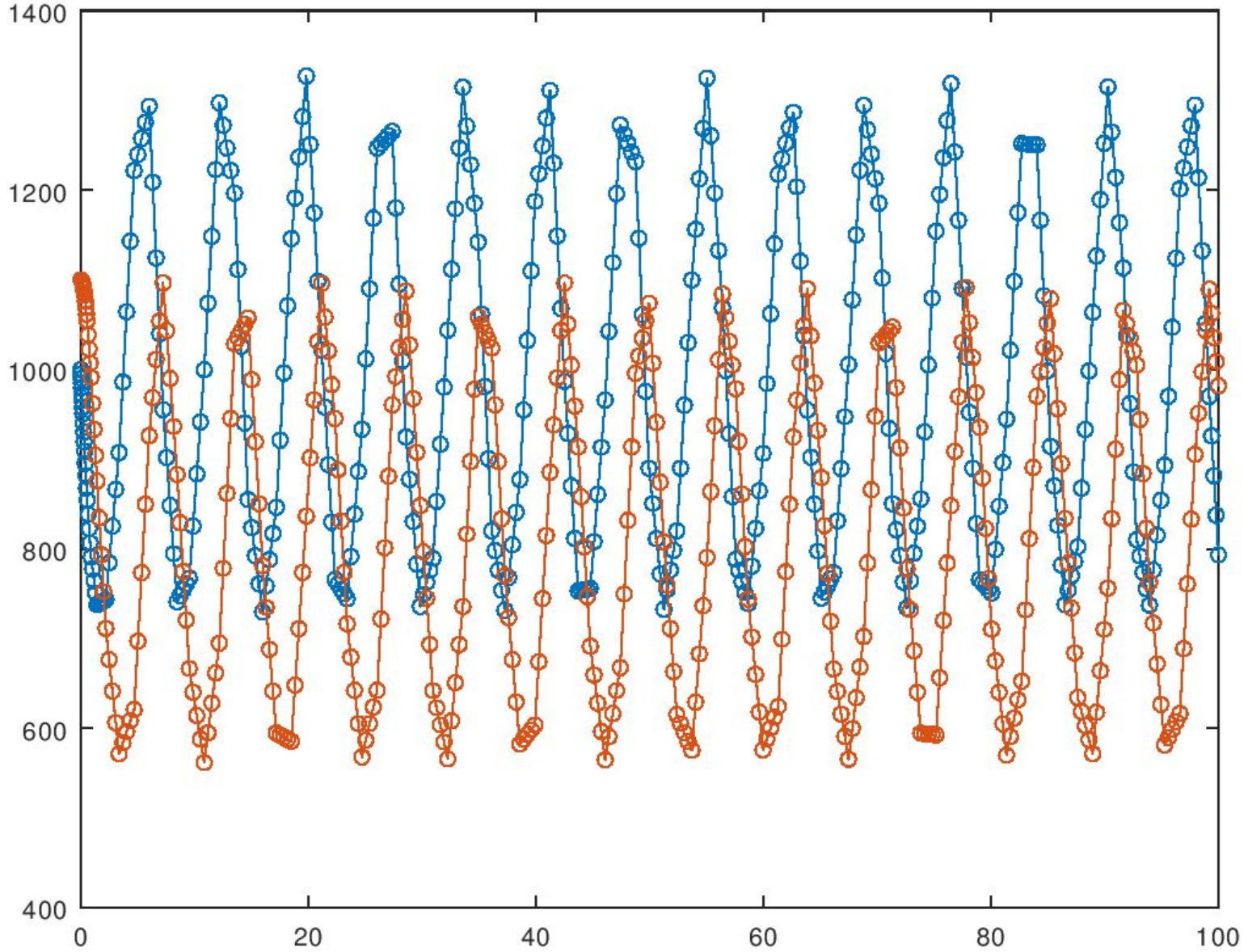
Солверы допускают решение систем обыкновенных дифференциальных уравнений, правая часть которых зависит от некоторых параметров ***p1, p2***, ... с известными значениями.

Способ обращения к солверам состоит в составлении файл функции со списком входных аргументов, соответствующих этим параметрам, формировании подфункции (или анонимной функции) вычисления правой части системы дифференциальных уравнений. Параметры для данной функции являются глобальными. Далее вызывается солвер.

Приведенная выше система дифференциальных уравнений Лотки—Вольтерры зависит от четырех параметров: *P, p, R* и г, и при многократном ее решении для различных значений параметров целесообразно написать соответствующую файл-функцию, вместо того, чтобы каждый раз изменять значения этих параметров. Если вектор-функция правой части дифференциального уравнения может быть записана в виде анонимной функции, то передача параметров может быть организована так:

```
function [T,Y] = LotVolPar(P,p,R,r)
% подфункция для вычисления правой части
системы,  зависящей от параметров
F =@(t,y) [P*y(1)-p*y(1)*y(2);-R*y(2) + r*y(1)*y(2)];
y0 = [1000; 1100]; % задание начальных условий
% вызов солвера
% построение графика решения
options = odeset('OutputFcn', @odeplot);
[T Y] = ode45(F, [0 100], y0,options);
endfunction
>> P=0.8;p=0.001;R=1;r=0.001;LVparl(P,p,R,r);
```

Выполните в командной строке этот оператор


1400
1200
1000
800
600
400
0
20
40
60
80
100

Если вектор-функция правой части дифференциального уравнения достаточно сложная, содержит несколько операторов и не может быть записана в виде анонимной функции, то применяют метод подфункций следующим образом. Рассмотрим на том же примере.
Текст программы ниже:

```
function [T,Y] = LotVolPar(P,p,R,r)
% Анонимная функция, вызывающая подфункцию
% зависящую от параметров
F =@(t,y) F1(t,y,P,p,R,r);
y0 = [1000; 1100]; % задание начальных условий
% вызов солвера
% построение графика решения
options = odeset('OutputFcn', @odeplot);
[T Y] = ode45(F, [0 100], y0,options);
endfunction

function z=F1(t,y,P,p,R,r)
% подфункция для вычисления правой части
системы,
% зависящей от параметров
  z=[P*y(1)-p*y(1)*y(2);-R*y(2) + r*y(1)*y(2)];
endfunction
```

Системы, не разрешенные относительно производной, дифференциально-алгебраические уравнения

До сих пор мы рассматривали задачу Коши для систем дифференциальных уравнений, разрешенных относительно производных

$$Y' = f(t,Y).$$

MATLAB позволяет решать системы дифференциальных уравнений, заданных в неявной форме

Важным частным случаем таких систем являются системы с матрицей

$$M(t,Y)Y'=F(t,Y),$$

Причем матрица $M(t, Y)$ может быть как невырожденной, так и вырожденной.

Для решения систем вида $F(t, Y, Y') = 0$ служит солвер ***ode15i***, а системы с матрицей могут быть решены любым из солверов.

Обратимся теперь к решению задачи Коши для дифференциальных уравнений вида ***F(t, Y, Y')=0***, не разрешенных относительно старшей производной, при помощи солвера ***ode15i***. Обращение к нему несколько отличается от случая рассмотренных выше солверов, поскольку во входных его аргументах должно быть задано значение ***Yp0*** производной решения в начальный момент времени $Y'(t_0)$:

[T, Y] = ode15i(odefun, interval, Y0, Yp0, options)

Смысл остальных его аргументов тот же самый: odefun— функция для вычисления вектор-функции ***F***; ***interval*** — отрезок, по которому производится интегрирование, или упорядоченный вектор значений независимой переменной, для которых следует найти решение; ***Y0*** — вектор начальных значений $Y(t_0)$; ***options***— управляющая структура.

Возникает вопрос, откуда взять начальное значение для производной, которое не может быть произвольным, поскольку в начальный момент времени t_0 требуется выполнение условия совместности $F(t_0, Y(t_0), Y'(t_0))=0$.Для приближенного удовлетворения этому условию служит функция ***decic***.

В достаточно общем случае обращение к ней выглядит следующим образом:

[Y0,Yp0] = decic(odefun, t0, Y0Init, Y0Flag, Yp0init, Yp0Flag)

и подразумевает, что заданы начальные приближения к $Y(t_0)$ **и** $Y'(t_0)$ соответственно в векторах ***Y0init u Yp0init***. Для запрета изменения некоторой компоненты решения или производной следует установить элемент вектора флага с номером компоненты в 1, а для разрешения в 0.

В задаче Коши вектор $Y(t_0)$ задан и требуется найти подходящее $Y'(t_0)$. Для этого следует задать флаг ***Y0init***, целиком состоящий из единиц. Если никакие компоненты векторов ***Y0init*** или ***Yp0init*** не фиксируются, то соответствующий флаг можно не задавать.
Возвращаемые значения ***Y0*** и ***Yp0*** приближенно удовлетворяют условиям совместности и могут служить входными аргументами солвера ***ode15i***.
Продемонстрируем использование солвера ***ode15i*** на примере уравнения Клеро:
Y=Y't+h(Y')

Известно, что ***y = Ct + h(C)*** есть общее решение этого уравнения, с ним мы и будем сравнивать получаемое приближенное решение. Рассмотрим задачу Коши для уравнения Клеро при $h(s)=e^{s}$, $y=y't+e^{y'}$, t *принадлежит* ***[0,5]*** с точным решением ***y=t+e***

Поскольку это дифференциальное уравнение первого порядка, то неизвестной является всего одна функция, и при обращении к ***decic*** в качестве флагов используются числа ***Y0Flag = 1*** и ***Yp0Flag = 0***

```
function klero
t0 = 0; % начальная точка отрезка интегрирования [0, 5]
Y0Init = exp(1); % заданное начальное условие
Y0Flag = 1 ; % начальное значение искомой функции не должно
% измениться
% при удовлетворении условиям совместности
Yp0Init = 0 ; % приближение для начального значения производной
Yp0Flag = 0 ; % начальное значение производной искомой функции
%  может измениться при удовлетворении условиям совместности
format long e
% Поиск начального значения производной, удовлетворяющего
% условиям совместности
[Y0, Yp0] = decic(@klerofun, t0, Y0Init, Y0Flag, Yp0Init, Yp0Flag)
% Решение уравнения Клеро на отрезке [0, 5]
[T, Y] = ode15i(@klerofun, [t0 5], Y0, Yp0) ;
% Построение графика приближенного решения
plot(T, Y, ' o ‘);
hold on
% Построение графика точного решения
exsol = inline ( ‘x + exp ( 1 ) ‘ ) ;
fplot ( exsol , [t  5])
legend('приближенное решение','точное решение')
function F = klerofun ( t , Y, Yp)
F' = Y - Yp*t - exp(Yp);
```

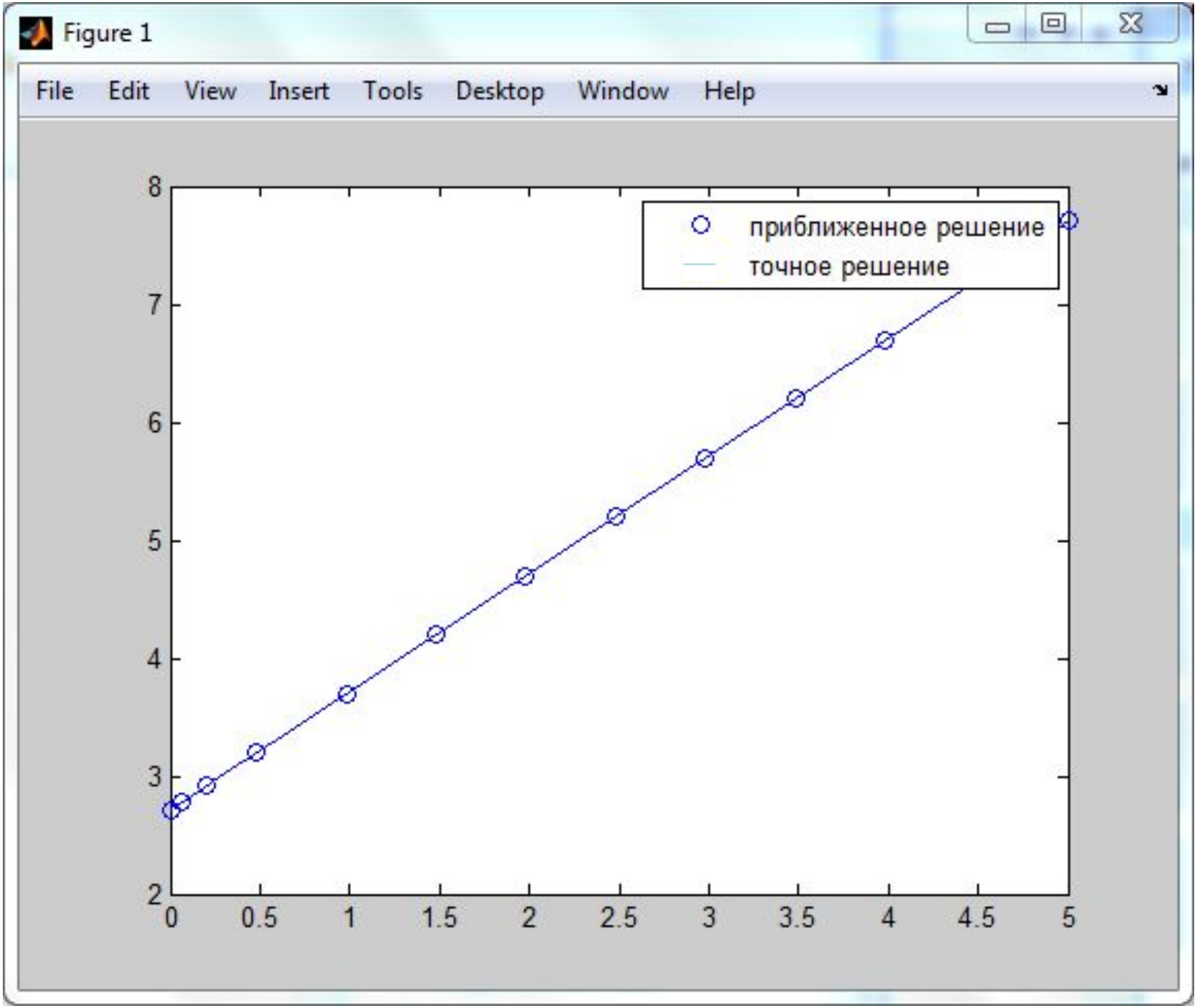


Figure 1
File Edit View Insert Tools Desktop Window Help
приближенное решение
точное решение
8
7
6
5
4
3
2
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5

Задание

- Написать программу решения дифференциальных уравнений:

$$y'' = \frac{y'}{x} + \frac{x^2}{y'},\ при\ \ y(0.1) = 0, y'(0.1) = 4$$ x_{max}=15

$$y'' = m\left(1 - y^2\right)y' - y,\ при\ \ y(0) = 0,\ y'(0) = 1,\ m = 50$$

$$x \in [0,1]$$

Записать в виде функции, зависящей от параметра m,