

9.4 ORDINARY DIFFERENTIAL EQUATIONS

Differential equations play a crucial role in science and engineering since they are in the foundation of virtually every physical phenomenon that is involved in engineering applications. Only a limited number of differential equations can be solved analytically. Numerical methods, on the other hand, can result in an approximate solution to almost any equation. Obtaining a numerical solution might not be simple task however. This is because a numerical method that can solve any equation does not exist. Instead, there are many methods that are suitable for solving different types of equations. MATLAB has a large library of tools that can be used for solving differential equations. To fully utilize the power of MATLAB, however, requires that the user have knowledge of differential equations and the various numerical methods that can be used for solving them.

This section describes in detail how to use MATLAB to solve a first-order ordinary differential equation. The possible numerical methods that can be used for solving such an equation are described in general terms, but are not explained from a mathematical point of view. This section provides information for solving simple, “nonproblematic” first-order equations. This solution provides the basis for solving higher-order equations and systems of equations.

An ordinary differential equation (ODE) is an equation that contains an independent variable, a dependent variable, and derivatives of the dependent variable. The equations that are considered here are of first order with the form

$$\frac{dy}{dx} = f(x, y)$$

where x and y are the independent and dependent variables, respectively. A solution is a function $y = f(x)$ that satisfies the equation. In general, many functions can satisfy a given ODE, and more information is required for determining the solution of a specific problem. The additional information is the value of the function (the dependent variable) at some value of the independent variable.

Steps for solving a single first-order ODE:

For the remainder of this section the independent variable is taken as t (time). This is done because in many applications time is the independent variable, and also to be consistent with the information in the **Help** menu of MATLAB.

Step 1: Write the problem in a standard form.

Write the equation in the form:

$$\frac{dy}{dt} = f(t, y) \quad \text{for } t_0 \leq t \leq t_f, \text{ with } y = y_0 \text{ at } t = t_0.$$

As shown above, three pieces of information are needed for solving a first order ODE: An equation that gives an expression for the derivative of y with respect to t , the interval of the independent variable, and the initial value of y . The solution is the value of y as a function of t between t_0 and t_f .

An example of a problem to solve is:

$$\frac{dy}{dt} = \frac{t^3-2y}{t} \quad \text{for } 1 < t < 3 \quad \text{with } y = 4.2 \text{ at } t = 1.$$

Step 2: Create a user-defined function (in a function file) or an anonymous function.

The ODE to be solved has to be written as a user-defined function (in a function file) or as an anonymous function. Both calculate $\frac{dy}{dt}$ for given values of t and y . For the example problem above, the user-defined function (which is saved as a separate file) is:

```
function dydt=ODEexpl(t,y)
dydt=(t^3-2*y)/t;
```

When an anonymous function is used, it can be defined in the Command Window, or be within a script file. For the example problem here the anonymous function (named ode1) is:

```
>> ode1=@(t,y) (t^3-2*y)/t
ode1 =
    @(t,y) (t^3-2*y)/t
```

Step 3: Select a method of solution.

Select the numerical method that you would like MATLAB to use in the solution. Many numerical methods have been developed to solve first-order ODEs, and several of the methods are available as built-in functions in MATLAB. In a typical numerical method, the time interval is divided into small time steps. The solution starts at the known point y_0 , and then by using one of the integration methods the value of y is calculated at each time step. Table 9-1 lists seven ODE solver commands, which are MATLAB built-in functions that can be used for solving a first-order ODE. A short description of each solver is included in the table.

Table 9-1: MATLAB ODE Solvers

ODE Solver Name	Description
ode45	For nonstiff problems, one-step solver, best to apply as a first try for most problems. Based on explicit Runge-Kutta method.
ode23	For nonstiff problems, one-step solver. Based on explicit Runge-Kutta method. Often quicker but less accurate than ode45.
ode113	For nonstiff problems, multistep solver.

Table 9-1: MATLAB ODE Solvers (Continued)

ODE Solver Name	Description
ode15s	For stiff problems, multistep solver. Use if ode45 failed. Uses a variable order method.
ode23s	For stiff problems, one-step solver. Can solve some problems that ode15s cannot.
ode23t	For moderately stiff problems.
ode23tb	For stiff problems. Often more efficient than ode15s.

In general, the solvers can be divided into two groups according to their ability to solve stiff problems and according to whether they use on-step or multistep methods. Stiff problems are ones that include fast and slowly changing components and require small time steps in their solution. One-step solvers use information from one point to obtain a solution at the next point. Multistep solvers use information from several previous points to find the solution at the next point. The details of the different methods are beyond the scope of this book.

It is impossible to know ahead of time which solver is the most appropriate for a specific problem. A suggestion is to first try ode45, which gives good results for many problems. If a solution is not obtained because the problem is stiff, trying the solver ode15s is suggested.

Step 4: Solve the ODE.

The form of the command that is used to solve an initial value ODE problem is the same for all the solvers and for all the equations that are solved. The form is:

`[t, y] = solver_name(ODEfun, tspan, y0)`

Additional information:

<code>solver_name</code>	Is the name of the solver (numerical method) that is used (e.g. ode45 or ode23s)
<code>ODEfun</code>	The function from Step 2 that calculates $\frac{dy}{dt}$ for given values of t and y . If it was written as a user-defined function, the function handle is entered. If it was written as an anonymous function, the name of the anonymous function is entered. (See the example that follows.)
<code>tspan</code>	A vector that specifies the interval of the solution. The vector must have at least two elements but can have more. If the vector has only two elements, the elements must be <code>[t0 tf]</code> , which are the initial and final points of the solution interval.

The vector `tspan` can have, however, additional points between the first and last points. The number of elements in `tspan` affects the output from the command. See `[t,y]` below.

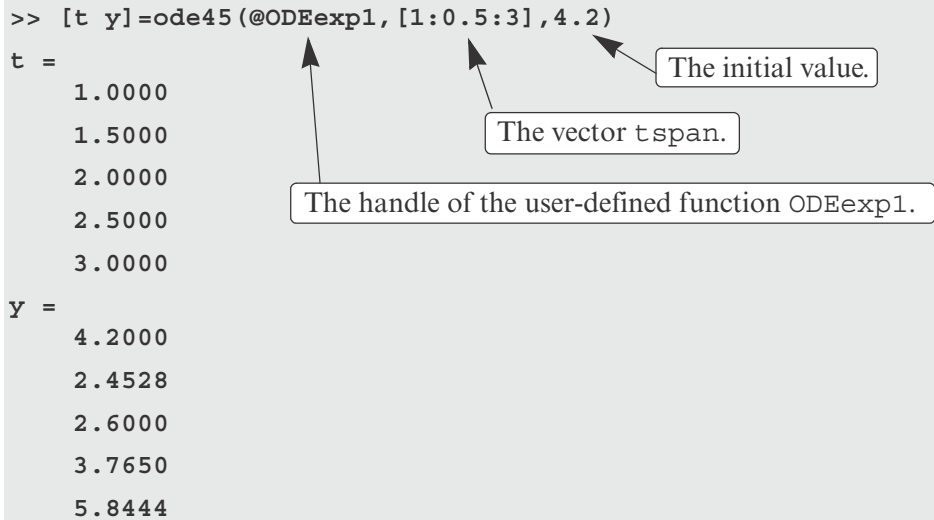
`y0` The initial value of y (the value of y at the first point of the interval).

`[t,y]` The output, which is the solution of the ODE. t and y are column vectors. The first and the last points are the beginning and end points of the interval. The spacing and number of points in between depends on the input vector `tspan`. If `tspan` has two elements (the beginning and end points), the vectors t and y contain the solution at every integration step calculated by the solver. If `tspan` has more than two points (additional points between the first and the last), the vectors t and y contain the solution only at these points. The number of points in `tspan` does not affect the time steps used for the solution by the program.

For example, consider the solution to the problem stated in Step 1:

$$\frac{dy}{dt} = \frac{t^3 - 2y}{t} \quad \text{for } 1 < t < 3 \quad \text{with } y = 4.2 \text{ at } t = 1,$$

If the ODE function is written as a user-defined function (see Step 2), then the solution with MATLAB's built-in function `ode45` is obtained by:



```
>> [t y]=ode45(@ODEexp1,[1:0.5:3],4.2)
```

`t =`

1.0000
1.5000
2.0000
2.5000
3.0000

`y =`

4.2000
2.4528
2.6000
3.7650
5.8444

Annotations:

- The handle of the user-defined function `ODEexp1`.
- The vector `tspan`.
- The initial value.

The solution is obtained with the solver `ode45`. The name of the user-defined function from Step 2 is `ODEexp1`. The solution starts at $t = 1$ and ends at $t = 3$ with increments of 0.5 (according to the vector `tspan`). To show the solution, the problem is solved again below using `tspan` with smaller spacing,