

Solving Ordinary Differential Equations with MATLAB

1. Introduction

2. What is an Ordinary Differential Equation?

Summary: What is an Ordinary Differential Equation?

Differential equations represent a functional relationship between two variables, an independent variable and a dependent variable.

dependent variable

$$\frac{dv}{dt} = -mg$$

independent variable

In order to solve a differential equation, you need to know at least one value of the dependent variable at some value of the independent variable. This is usually provided as an initial condition or boundary condition.

value of dependent variable

$$v(0) = 100$$

value of independent variable

The solution to a differential equation is a function, not a single number. Methods of calculus can determine the solution exactly, but they are beyond the scope of this course. You can also solve ODEs approximately using numerical methods in MATLAB, which results in vectors of values representing the independent and dependent variables.

Exact Solution

$$v(t) = 100 - mgt$$

Numerical Solution

t	v
0	100
0.2	97.02
0.4	94.04
0.6	91.06
0.8	88.08
⋮	⋮

3. Solving ODEs Numerically

Summary: Solving ODEs Numerically

MATLAB ODE Solver Syntax

ode45 with two outputs

```
>> [tSol,ySol] = ode45(@odefun,tspan,y0)
```

Outputs		Inputs	
tSol	A vector containing the values of the independent variable at which the numerical solution was computed.	@odefun	A function handle to the ODE function.
ySol	A vector containing the computed value of the dependent variable for each time in tSol .	tspan	A vector containing the initial and final values for the independent variable.
		y0	The initial value of the dependent variable.

ode45 with one output

```
>> sol = ode45(@odefun,tspan,y0)
```

Outputs		Inputs	
sol	A solution structure containing information about the solution, such as the solver and evaluation points. Use deval to evaluate the solution sol at any point in tspan .	@odefun	A function handle to the ODE function.
		tspan	A vector containing the initial and final values for the independent variable.
		y0	The initial value of the dependent variable.

Structure of an ODE function

The *ODE function* is a MATLAB function that represents the ODE for use in a MATLAB ODE solver, like ode45 .

$$\frac{dy}{dt} = y \left(1 - \frac{y}{K}\right),$$

where $K = 1000$

The function declaration for an ODE function begins with the keyword **function**, followed by a single output, the function name, and two inputs (independent then dependent variables).

```
odefun
1. function dydt = odefun(t,y)
2. K = 1000;
3. dydt = y*(1 - y/K);
4. end
```

The function body implements the ODE you are trying to solve.

```
odefun
1. function dydt = odefun(t,y)
2. K = 1000;
3. dydt = y*(1 - y/K);
4. end
```

The function ends with the keyword **end**.

```
odefun
1. function dydt = odefun(t,y)
2. K = 1000;
3. dydt = y*(1 - y/K);
4. end
```

4. Solving Systems of ODEs Numerically

Summary - Solving Systems of ODEs Numerically

You can solve a system of differential equations using `ode45` just like you would for a single differential equation. The main differences are the input `Y0` and the output `YSol`. The size of `Y0` and `YSol` depend on the number of dependent variables in your system.


```
>> [tSol,YSol] = ode45(@odefun,tspan,Y0)
```

Outputs		Inputs	
tSol	A vector containing the values of the independent variable at which the numerical solution was computed.	@odefun	A function handle to the derivative function.
YSol	A matrix containing the computed values of the dependent variables for each time in tSol. Each column of the matrix corresponds to one of the dependent variables.	tspan	A vector containing the initial and final values for the independent variable.
		Y0	A vector containing the initial values of the dependent variables.

The ODE Function for Systems of ODEs

To write an ODE function for a system of ODEs, the system is implemented as a vector-valued ODE.

$$\begin{aligned}\frac{dx}{dt} &= z + 4 \\ \frac{dz}{dt} &= -3x\end{aligned}$$



$$\mathbf{Y} = \begin{bmatrix} x \\ z \end{bmatrix}$$

$$\frac{d\mathbf{Y}}{dt} = \begin{bmatrix} \frac{dx}{dt} \\ \frac{dz}{dt} \end{bmatrix} = \begin{bmatrix} z + 4 \\ -3x \end{bmatrix}$$

The function declaration remains unchanged from the ODE function for a single ODE. However, the output `dYdt` and the second input `Y` are now column vectors.

Index into `Y` to extract each dependent variable.

Implement the system of ODEs by defining the derivative functions of each dependent variable.

odefun

```
1. function dYdt = odefun(t,Y)
2. x = Y(1);
3. z = Y(2);
4. dxdt = z + 4;
5. dzdt = -3*x;
6. dYdt = [dxdt;dzdt];
7. end
```

odefun

```
1. function dYdt = odefun(t,Y)
2. x = Y(1);
3. z = Y(2);
4. dxdt = z + 4;
5. dzdt = -3*x;
6. dYdt = [dxdt;dzdt];
7. end
```

Construct the output variable $dYdt$ as a column vector of the derivative functions.

odefun

```
1. function dYdt = odefun(t,Y)
2. x = Y(1);
3. z = Y(2);
4. dxdt = z + 4;
5. dzdt = -3*x;
6. dYdt = [dxdt;dzdt];
7. end
```

5. Solving Higher-Order ODEs Numerically

Summary - Solving Higher-Order ODEs Numerically

Provided you have rewritten your N^{th} -order ODE as a system of first-order ODEs, you can find its solution in the same manner any system of first-order ODEs is solved.

Implementing Higher-Order ODEs

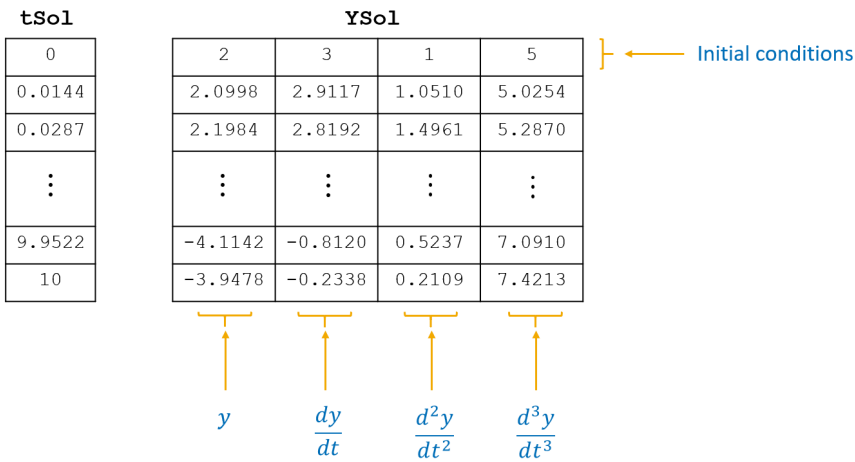
1. Rewrite N^{th} -order ODE as a system of first-order ODEs.
2. Write ODE function for the system of ODEs implemented as a vector-valued ODE.
3. Define the initial conditions Y_0 and the values of the independent variable, $tRange$ at which the solution must be computed.
4. Solve the system of ODEs using a MATLAB ODE solver like `ode45`.

$$\begin{aligned} \frac{d^2y}{dt^2} + 5 \frac{dy}{dt} + 2y &= 0 \\ \frac{dy}{dt} &= Y_2 \\ \frac{d^2y}{dt^2} &= \frac{dY_2}{dt} \\ y &= Y_1 \end{aligned} \quad \Rightarrow \quad \begin{aligned} \frac{dY_1}{dt} &= Y_2 \\ \frac{dY_2}{dt} &= -5Y_2 - 2Y_1 \end{aligned} \quad \Rightarrow \quad \begin{aligned} \mathbf{Y} &= \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \\ \frac{d\mathbf{Y}}{dt} &= \begin{bmatrix} \frac{dY_1}{dt} \\ \frac{dY_2}{dt} \end{bmatrix} = \begin{bmatrix} Y_2 \\ -5Y_2 - 2Y_1 \end{bmatrix} \end{aligned}$$

The Solution Matrix for Higher-Order ODEs

The matrix containing the computed values of the dependent variable, `YSol`, also contains values of the derivatives of the dependent variable. For an N^{th} -order ODE, `YSol` will contain the dependent variable and up to $N-1$ of its

derivatives. The vector `tSol` still contains the values of the independent variable.



6. Next Steps