

## 12 Differential equations

Differential equations are equations where we are solving for a function rather than a variable. For example, instead of solving for  $x$  in a normal algebraic equation such as

$$a \cdot x = b$$

we now solve for a function  $y(t)$  in a differential equation such as

$$y(t) = a \cdot \frac{\partial y(t)}{\partial t}.$$

Differential equations are closely related to integral equations. The differential and the integral operators are the inverse of each other. So for every differential equation there is one corresponding integral equation. For the above example, this would be

$$\int y(t) dt = a \cdot y(t).$$

We'll usually work with differential equations rather than integral equations because they are more intuitive most of the time.

Let's discuss a few examples of differential equations. For these differential equations we can write down exact solutions.

**Example 1:** The following differential equation pops up in a variety of places in biology and physics:

$$N(t) = a \cdot N'(t).$$

It says that the growth of the function,  $N'(t)$ , is proportional to the value of the function  $N(t)$ . The solution is an exponential:

$$N(t) = N_0 \cdot e^{a \cdot t}$$

Here,  $N_0$  is an initial value that is determined by the boundary conditions.

**Example 2:** The first example leads to an exponentially growing or decreasing function. Another class of differential equations leads to periodic solutions. For example

$$m \cdot x''(t) = -k \cdot x(t)$$

has a solution

$$x(t) = A \cdot \cos(\omega t + \phi) \quad \text{with} \quad \omega^2 = \frac{k}{m}.$$

This is the solution for a harmonic oscillator, a very common problem in physics.

So, let's go back to the initial conditions. What are these values  $N_0$ ,  $A$  and  $\phi$ ? They are given to you as part of the problem description. For example, in the first example, if you think of  $N$  as the number of creatures in a biology model, then  $N_0$  is the number of creatures at the beginning of the experiment. Similarly, in the second example, the initial position and velocity of your pendulum determine the constants  $A$  and  $\phi$ .

The unknown function is not entirely specified by the differential equation itself. We also need to define certain boundary conditions. The nature of the boundary conditions varies depending on the problem. They can be as simple as requiring that the function has a certain value at  $t = 0$ . But they can also be complex algebraic equations. Different boundary conditions lead to qualitatively different problems and solutions. There are two main kinds of differential equations.

- The first kind has the boundary conditions specified at one point.
- The second kind has boundary conditions for two points

In this course we will focus on the first kind.

## 12.1 Euler Method

The simplest method to solve a differential equation of the form

$$y'(t) = F(y, t)$$

is the Euler method. We start from an initial condition consisting of two values,  $t_0$  and  $y_0 = y(t_0)$ . We then integrate the equation forward in time, one timestep  $dt$  at a time. Often the timestep is also denoted by the letter  $h$ . The size of the timestep determines the precision of the scheme. The smaller the timestep the higher the accuracy but the more calculations we have to do. This statement is true not only for the Euler method but for all ODE (ordinary differential equation) solvers.

To perform an Euler step, we evaluate the derivate of the function  $y(t)$  at the beginning of the timestep. For the first timestep,  $t$  is  $t_0$ , for the second timestep  $t_0 + dt$ , and so on. We then multiply the derivative with the timestep  $dt$  and add it to the initial value  $y_0$ . If we label subsequent steps with the index  $n$ , then the Euler method can be written as

$$y_{n+1} = y_n + dt \cdot F(y_n, t_n).$$

Note that you can think of the the function  $y$  as either a scalar function or a vector function. The Euler method (and any other ODE solver) works exactly the same in both cases. In other words, it works in multiple dimensions as well as in just one. This can be used to convert an ODE with a second order derivative of  $y$  into a system of first order derivatives in vector notation. As an example, suppose we want to solve the ODE

$$y'' = F(y)$$

then we can rewrite this as a vector equation and first derivatives as

$$\begin{pmatrix} y' \\ y'' \end{pmatrix} = \begin{pmatrix} y' \\ F(y) \end{pmatrix}$$

and thus

$$\begin{pmatrix} y \\ y' \end{pmatrix}' = \begin{pmatrix} y' \\ F(y) \end{pmatrix}$$

or equally

$$Y' = G(Y)$$

where we introduced new variables  $Y$  and the function  $G$ .

We now perform an error analysis on the Euler method. We do this to find out how well the Euler method does, how large the error is and how it depends on the timestep  $dt$ . So, let's go back to our generic example

$$y'(t) = F(y, t)$$

and expand the function  $y$  as a Taylor series around the  $t = t_0$

$$\begin{aligned} y(t) &= y_0 + (t - t_0) \cdot \frac{\partial y}{\partial t} + \frac{1}{2}(t - t_0)^2 \cdot \frac{\partial^2 y}{\partial t^2} + \dots \\ &= y_0 + dt \cdot F(y, t) + \frac{1}{2}dt^2 \cdot \frac{\partial^2 y}{\partial t^2} + \dots \end{aligned}$$

The Euler method gets the first two terms right. The error after one timestep is therefor of the order

$$E \sim \frac{1}{2}dt^2 \cdot \frac{\partial^2 y}{\partial t^2}$$

plus higher order terms. The higher order terms will be small if  $dt$  is small and the dominant error term will be of order  $dt^2$ .

Suppose we want to integrate a system forward in time for some finite time  $T$ . Then the number of timesteps  $N$  depends on the size of the timestep  $dt$ :

$$N = \frac{T}{dt}$$

The smaller the timestep, the more timesteps we need to take. Thus, the error of the Euler method after a finite time  $T$  is now not of order  $dt^2$  anymore, but of order  $dt$ . This is an important result. The Euler method is a first order method. If we reduce the timestep by a factor of two, then the error will be smaller by a factor of two. This is good because we eventually converge to the correct solution if we only make the timestep small enough. However, we might hope to find a better method that is more accurate than the Euler method and converges faster, i.e. quadratically. This is important because in most situations the evaluation of the function  $F$  is very computationally expensive and we want to minimize the number of evaluations. The Euler method has one function evaluation per timestep.

## 12.2 Runge-Kutta Methods

As we indicated before, we might be able to cancel out higher order terms in the expansion of  $y$ . This would lead us to a high order scheme which is better and faster than the simple Euler Methods. A class of higher order methods that is widely used is the class of Runge-Kutta Methods. These methods use the Euler method to make predictions of  $y$  during the timestep and then use this information to improve the result at the end.

The mid-point method, or second-order Runge-Kutta method, is one such method. The idea is simple. Let's first take half an Euler step, to get an estimate of the function  $F$  in the middle of the timestep. Then use this estimate to advance  $y$  from the beginning of the timestep to the end. In terms of equations, it can be written as follows (using the same notation as above):

$$\begin{aligned} k_1 &= dt \cdot F(y_n, t_n) \\ k_2 &= dt \cdot F\left(y_n + \frac{1}{2} \cdot k_1, t_n + \frac{1}{2}dt\right) \\ y_{n+1} &= y_n + k_2 \end{aligned}$$

Let's do an error analysis for this method. The Taylor expansion of the function  $y$  to third order is

$$y(t) = y_0 + (t - t_0) \cdot \frac{\partial y}{\partial t} + \frac{1}{2}(t - t_0)^2 \cdot \frac{\partial^2 y}{\partial t^2} + \frac{1}{3}(t - t_0)^3 \cdot \frac{\partial^3 y}{\partial t^3} + \dots$$

An integration step of the mid-point method gives

$$\begin{aligned} y_{n+1} &= y_n + k_2 \\ &= y_n + dt \cdot F\left(y_n + \frac{1}{2} \cdot k_1, t_n + \frac{1}{2}dt\right) \\ &= y_n + dt \cdot F\left(\underbrace{y_n + \frac{1}{2} \cdot dt \cdot F(y_n, t_n)}_{=y_{n+\frac{1}{2}} + O(dt^2)}, t_n + \frac{1}{2}dt\right) \\ &= y_n + dt \cdot F\left(\underbrace{y_n + \frac{1}{2} \cdot dt \cdot F(y_n, t_n)}_{=F_{n+\frac{1}{2}} + O(dt^2)}, t_n + \frac{1}{2}dt\right) \end{aligned}$$

We've seen this before. This is the central difference scheme from last lecture (ignoring the higher order error term). The central difference scheme was used to estimate the derivative of a function. It is symmetric with respect to a reflection around the mid-point. This makes it higher order (2nd). Here, this is what makes the mid-point method second order accurate. We now have an error of order  $O(dt^3)$  after one timestep. If the error of a scheme is  $O(dt^{n+1})$  after one timestep, then we call the scheme  $n$ -th order.

The most commonly used Runge-Kutta method is the fourth order Runge-Kutta Method, or RK4. It uses four function evaluations and can be written in a similar way as the mid-point method.

$$\begin{aligned}
 k_1 &= dt \cdot F(y_n, t_n) \\
 k_2 &= dt \cdot F\left(y_n + \frac{1}{2} \cdot k_1, t_n + \frac{1}{2} dt\right) \\
 k_3 &= dt \cdot F\left(y_n + \frac{1}{2} \cdot k_2, t_n + \frac{1}{2} dt\right) \\
 k_4 &= dt \cdot F(y_n + k_3, t_n + dt) \\
 y_{n+1} &= y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4
 \end{aligned}$$

As one can show (we won't) the error after one timestep is  $O(h^5)$ , thus it is indeed a fourth order method.

There are many different Runge-Kutta scheme. They are usually written in block form. For example, the coefficients for RK4 can be summarized as

$$\begin{array}{c|ccc}
 0 & & & \\
 \frac{1}{2} & \frac{1}{2} & & \\
 \frac{1}{2} & & \frac{1}{2} & \\
 1 & & & 1 \\
 \hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
 \end{array}$$

Note that the 0's are not printed in the block form. Similarly, the Euler method can be summarized as

$$\begin{array}{c|c}
 0 & \\
 \hline
 & 1
 \end{array}$$

And the midpoint method can be summarized as

$$\begin{array}{c|cc}
 0 & & \\
 \frac{1}{2} & \frac{1}{2} & \\
 \hline
 & 0 & 1
 \end{array}$$

End of lecture 9.