Numerical Solution of Differential Equations

1. Ordinary Differential Equation

Ordinary differential equations are equations of the form

$$\frac{dx}{dt} = f(x,t) \text{ with } x(0) = x_0 \qquad (1)$$

On the left hand side is the derivative of the dependent variable x with respect to the independent variable t. On the right hand side, there is a function that may depend on both x and t. The independent variable t often represents time. In contrast to discrete time equations of the form $x_{t+1} = f(x_t)$, where time t is discrete ($t = 1, 2, \ldots$), the independent variable t in Equation (1) is a continuous variable, that is, it takes on real values, for instance, $t \ni [0, \infty]$. In addition, we prescribe the initial value at time 0, namely x_0. (The initial condition could be stated for some other time but time 0 is quite commonly used). A differential equation is called ***ordinary*** if it involves only one independent variable.

Many differential equations cannot be solved exactly. Numerical methods have been developed to approximate solutions. Numerical analysis is a field in mathematics that is concerned with developing approximate numerical methods and assessing their accuracy, for instance for solving differential equations. We will discuss the most basic method such Taylor and Euler methods.

1.1 Euler's Method

To find numerical solution to the initial value problem

$$\frac{dy}{dx} = f(x,y) \qquad y(0) = y_0 \qquad (4)$$

Using Euler's method we have the following consideration:

$$\begin{array}{ll}
x_1 = x_0 + h & y_1 = y_0 + h \cdot f(x_0, y_0) \\
x_2 = x_1 + h & y_2 = y_1 + h \cdot f(x_1, y_1) \\
x_3 = x_2 + h & y_3 = y_2 + h \cdot f(x_2, y_2) \\
\vdots & \vdots \\
\vdots & \vdots \\
\vdots & \vdots
\end{array}$$

Exercise 1:

Apply Euler's method to approximate the solution of the initial value problem

$$\frac{dy}{dx} = 2y \quad \text{with} \quad y(0) = 5 \tag{5}$$

We know what the solution of equation (5) is, namely $y = 5\exp(2x)$. We numerically solve equation (5) using Euler's method with h=0.1 in the time interval [0, 0.5], and then check how well this method performs. We have $f(y) = 2y$. Then

$$\begin{aligned}
x_0 &= 0 \\
x_1 &= x_0 + h = 0 + 0.1 = 0.1 \\
x_2 &= x_1 + h = 0.1 + 0.1 = 0.2 \\
x_3 &= x_2 + h = 0.2 + 0.1 = 0.3 \\
x_4 &= x_3 + h = 0.3 + 0.1 = 0.4 \\
x_5 &= x_4 + h = 0.4 + 0.1 = 0.5
\end{aligned}$$

And

$$\begin{aligned}
y_0 &= 5 \\
y_1 &= y_0 + hf(y_0) = 5 + \underbrace{(0.1)}_{h}\underbrace{(2)(5)}_{f(y_0)} = 6 \\
y_2 &= y_1 + hf(y_1) = 6 + (0.1)(2)(6) = 7.2 \\
y_3 &= y_2 + hf(y_2) = 7.2 + (0.1)(2)(7.2) = 8.64 \\
y_4 &= y_3 + hf(y_3) = 8.64 + (0.1)(2)(8.64) = 10.368 \\
y_5 &= y_4 + hf(y_4) = 10.368 + (0.1)(2)(10.368) = 12.4416
\end{aligned}$$

We summarize this in the following table. If h=0.1, then

x	y	Exact	Difference
0	5	5	0
0.1	6	6.107014	0.107014
0.2	7.2	7.459123	0.259123
0.3	8.64	9.110594	0.470594
0.4	10.368	11.1277	0.759705
0.5	12.4416	13.59141	1.149809

The third column contains the exact values, $y = 5\exp(2x)$. The last error contains the absolute error after each step, computed as |y-Exact|. We see that when h=0.1, the numerical approximation is not very good after five steps. If we repeat the same approximation with a smaller value for h, say h=0.01, the following table results for the first five steps:

X	y	Exact	Difference
0	5	5	0
0.01	5.1	5.101007	0.001007
0.02	5.202	5.204054	0.002054
0.03	5.30604	5.309183	0.003143
0.04	5.412161	5.416435	0.004275
0.05	5.520404	5.525855	0.005451

Doing five steps only gets us to x=0.05. We can do more steps until we reach x=0.5. We find that the final point will be:

X	y	Exact	Difference
0.5	13.45794	13.59141	0.133469

Choosing a smaller value for h resulted in a better approximation at x=0.5 but also required more steps. One source of error in the approximation comes from the approximation itself. Another source comes from rounding errors when we implement the approximation on a computer. It is therefore not necessarily the case that smaller values of h always improve the approximation.

Exercise 1: Simple first order ODE

Consider the following initial value problem:

$\frac{dx}{dt} = t^2 + t$ With the initial condition: $x(0) = 0.5$

Solution:

```
t_in=0;
t_fin=2;
nsteps=10;
dt=(t_fin-t_in)/nsteps;
t(1)=t_in;
x(1)= 0.5;
for n=1:nsteps
t(n+1)=t(n)+dt
x(n+1)=x(n)+dt*(t(n)^2+t(n))
end
plot(t,x),xlabel('Time (t)'),ylabel('x(t)')
```

The result is in Figure (1)

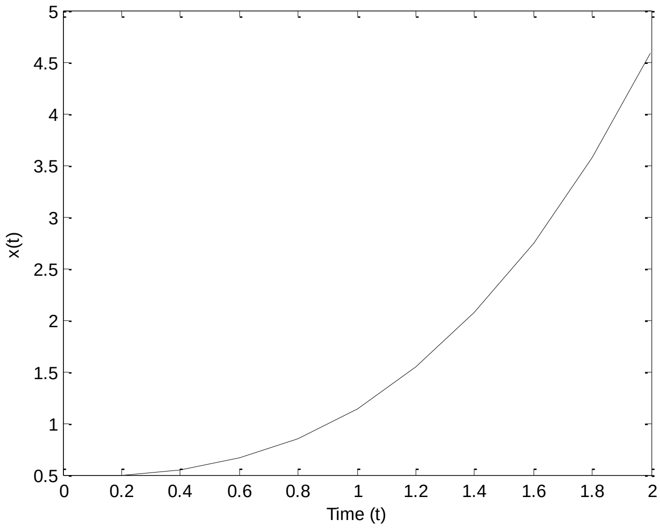

Figure 1: Euler method used to solve exercise1

This ODE can be analytically integrated to get the true solution:

$$x(t) = \frac{t^3}{3} + \frac{t^2}{2} + 0.5$$

Applying the explicit Euler's scheme with a step size h = 0.2 we get:

$$x(0.2) = x(0) + 0.2f(0)$$

That is

$$x(0.2) = 0.5 + 0 = 0.5$$

The true solution from is $x(0.2) = 0.52267$

The relative error (e_r) expressed in percent is

$$e_r = \left|\frac{True - approximation}{True}\right| \times 100$$

$$e_r = \left|\frac{0.5 - 0.52267}{0.5}\right| \times 100 = 4.5\%$$

The calculations results are plotted in Figure 2 showing the true and the approximate value for h=0.2. Although the general trend of the true and the approximate values is the same, the error is large. One way to reduce this error is by choosing a smaller step size. Figure 2 shows the solution when the step size is halved i.e. h=0.1. Since the Euler's method is first order, the global error is halved O (h /2) while the local error is quartered O (h2/4). To get acceptable levels of errors the step size has to be further reduced to very low values. This will however considerably increase the computational time since it will take a larger number of iterations for each step. Nevertheless the Euler's method because of its simplicity and easiness for implementation is still attractive for many engineering problems.

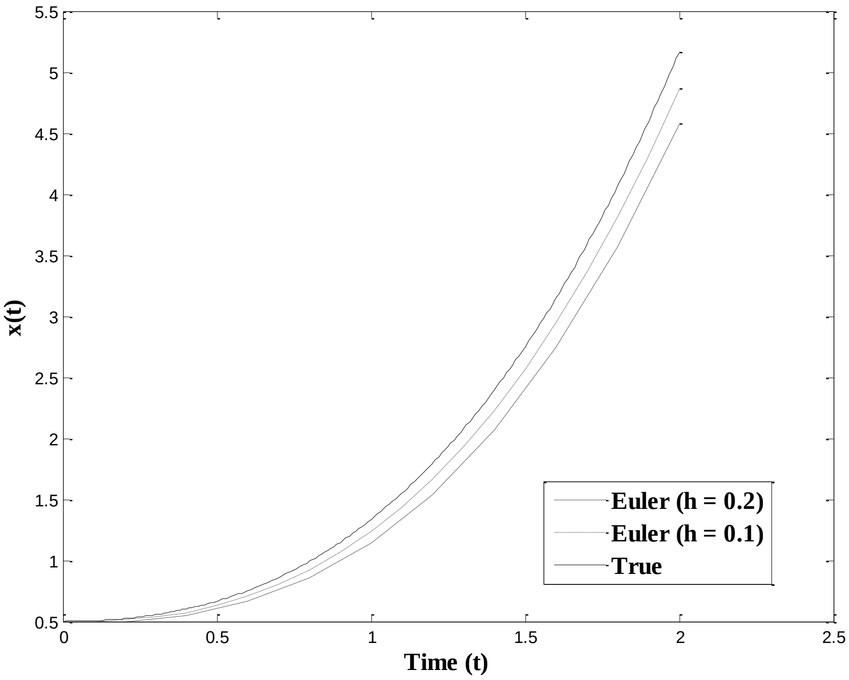

Figure 2: Euler method used to solve exercise 1

Exercise 2:

Write a program using Euler's method to solve the differential equation

$$\frac{dh}{dt} = 0.5 - h^{0.75}$$ Given the initial condition $h_o = 2.5$ m

In order to solve a particular differential equation, we need to define the step size dt from the initial and final t values t_in and t_fin, and the number of steps nsteps.

The solution is returned in an array h.

```
t_in=0; t_fin=5;
nsteps=10;
dt=(t_fin-t_in)/nsteps;
h0=2.5; t(1)=t_in;
h(1)=h0;
for n=1:nsteps
t(n+1)=t(n)+dt
h(n+1)=h(n)+dt*(0.5-h(n)^0.75)
end
plot(t,h,'k-'),xlabel('t'),ylabel('h')
```

The result is in Figure (3)

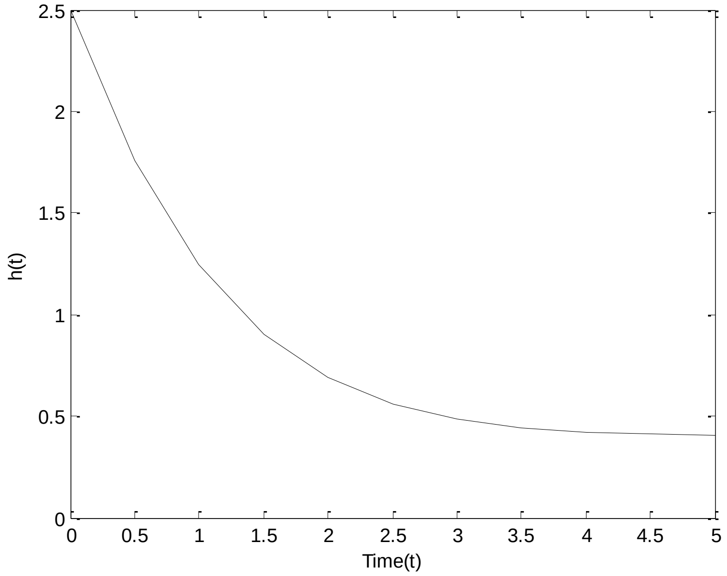

Figure 3: Euler method used to solve Exercise 2