Differential Equations and Taylor's Theorem

Recall:

Theorem 1 (Taylor's Theorem)**.** *For a function $x(t)$ that is $n+1$ times continuously differentiable on an interval containing t_0 and t_0+h, we have*

$$x(t_0+h) = x(t_0) + x'(t_0)\cdot h + \tfrac{1}{2}x''(t_0)\cdot h^2 + \tfrac{1}{3!}x'''(t_0)\cdot h^3 + \cdots + R_n$$

Euler's Method

Euler's Method is the first-degree Taylor approximation of the solution of the first-order initial value problem $\begin{cases} x'(t) = f(t,x) \\ x(t_0) = x_0 \end{cases}$.

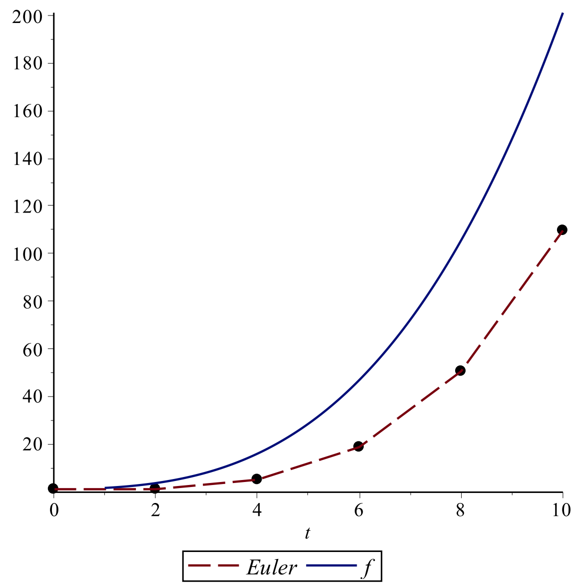
We start the approximation using $x_1 = x(t_0+h) = x(t_0) + x'(t_0)\cdot h$, so

$$\begin{cases} t_1 &= t_0 + h \\ x_1 &= x_0 + f(t_0,x_0)\cdot h \end{cases}$$

And iterate to make a general formula for $n = 1..N$ with $N = (b-a)/h$:

$$\begin{cases} t_n &= t_{n-1} + h \\ x_n &= x_{n-1} + f(t_{n-1},x_{n-1})\cdot h \end{cases} \qquad (1)$$
$$(t_0,x_0) = (a,x(a))$$

etc. The graph below shows Euler's method applied to the IVP $x' = t\sqrt[3]{x}$ with $x(0) = 1$ and $b = 10$.



Second Degree Method

Since $x'(t) = f(t,x)$, then $x''(t) = \frac{d}{dt}f(t,x(t))$. We use the second-degree Taylor approximation

$$x(t_0+h) = x(t_0) + x'(t_0)\cdot h + \tfrac{1}{2}x''(t_0)\cdot h^2$$

so that the new approximation uses $x_1 = x(t_0+h) = x(t_0) + x'(t_0)\cdot h + \frac{1}{2}x''(t_0)\cdot h^2$; that is,

$$x_1 = x_0 + f(t_0,x_0)\cdot h + \tfrac{1}{2}f_t(t_0,x_0)\cdot h^2$$

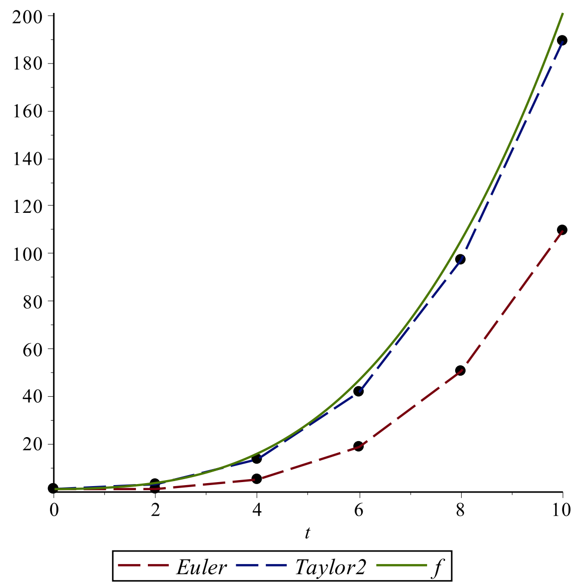
which gives

$$\begin{cases} t_1 &= t_0 + h \\ x_1 &= x_0 + f(t_0,x_0)\cdot h + \tfrac{1}{2}f_t(t_0,x_0)\cdot h^2 \end{cases}$$

Once again, iterate to make a general formula for $n = 1..N$ with $N = (b-a)/h$:

$$\begin{cases} t_n &= t_{n-1} + h \\ x_n &= x_{n-1} + f(t_{n-1}, x_{n-1}) \cdot h + \frac{1}{2} f_t(t_{n-1}, x_{n-1}) \cdot h^2 \end{cases} \qquad (2)$$
$$(t_0, x_0) = (a, x(a))$$

etc. (Or, if given N, then $b = a + N h$.) The graph below shows a second degree Taylor method applied to the IVP $x' = t\sqrt[3]{x}$ with $x(0) = 1$ and $b = 10$.



Maple Code to Generate the Next Point

```
EulerStep := proc(f,x_0,y_0,h,n)
    # f:y'(x,y); h:Delta_x; n:num steps
    local x0, x1, y0, y1, i;
    x0 := x_0;
    y0 := y_0;
    for i to n do
        x1 := x0 + h;
        y1 := y0 + f(x0,y0)*h;
        x0,y0 := x1,y1;
        end do;
    return([x0,y0])
    end proc:
```

```
Taylor2Step := proc(f,fp,x_0,y_0,h,n)
    # f:y'(x,y); fp:y''(x,y); h:Delta_x; n:num steps
    local x0, x1, y0, y1, i;
    x0 := x_0;
    y0 := y_0;
    for i to n do
        x1 := x0 + h;
        y1 := y0 + f(x0,y0)*h + (1/2)*fp(x0,y0)*h^2;
        x0,y0 := x1,y1;
        end do;
    return([x0,y0])
    end proc:
```