Euler's Method (§1.4)

EULER'S METHOD (Section 1.4)

```
> restart;
> with( plots ):
> with( DEtools ):
```

The direction fields introduced in the previous section provide a natural way to construct approximate solutions to an initial value problem (IVP). This method, which is commonly called Euler's Method, is not implemented in Maple. As we see the ideas behind this method we will be able to construct our own implementation of Euler's Method. This procedure will then be used in a variety of problems.

Objectives:

- to motivate and understand Euler's method
- to develop a Maple implementation, `euler`, of Euler's method
- to use `euler` to investigate new problems

New Maple Commands:

`->`	– the arrow operator (used for defining functions in Maple)
`for .. do .. od`	– the main Maple loop structure (see `?for` for complete details)
`display`	– combines several plots into a composite plot

Note that while the default numerical method used in `DEplot` is a fourth-order Runge–Kutta method, three Euler-type methods are also available: Euler's method ('euler'), backward Euler's method ('backeuler'), and improved Euler's method ('impeuler'). These can be specified using the `method=`; please see the on-line help for `DEtools[options]` for additional details.

The basic idea behind Euler's method is to use the slope at the current point – which is KNOWN (why?) – to determine the direction in which the solution should move. The next point on the approximate solution is found by taking a step, with a specified stepsize (h) in this direction. The process now repeats with the new current point.

EXAMPLE 1 (p. 25)

Use Euler's Method with stepsize $h = 0.1$ to approximate the solution to the initial value problem

```
> ODE := diff( y(x), x ) = x * sqrt( y(x) );
```

$$ODE := \frac{\partial}{\partial x}\mathrm{y}(x) = x\sqrt{\mathrm{y}(x)}$$

```
> IC := y(1) = 4;
```

$$IC := \mathrm{y}(1) = 4$$

at the point $x = 1.1$, 1.2, 1.3, 1.4, and 1.5.

Following the algorithm developed in the text, we have must identify the function, f, that is the right-hand side of the differential equation

```
> f := (x,y) -> x*sqrt(y);
```

$$f := (x, y) \to x\,\mathrm{sqrt}(y)$$

the stepsize

```
> h := 0.1;
```

$$h := .1$$

the number of steps to be taken

```
> N := 5;
```

$$N := 5$$

and the initial condition

```
> X[0] := 1;
```

$$X_0 := 1$$

```
> Y[0] := 4;
```

$$Y_0 := 4$$

Note that the function f is entered as a function, not an expression.

The slope at $x = 1$ is

```
> f( X[0], Y[0] );
```

$$2$$

so the approximate value of the solution at the first point is computed to be

```
> X[1] := X[0] + h;
```

$$X_1 := 1.1$$

```
> Y[1] := Y[0] + h*f( X[0], Y[0] );
```

$$Y_1 := 4.2$$

Repeating the same steps four more times yields:

```
> X[2] := X[1] + h;
```

$$X_2 := 1.2$$

```
> Y[2] := Y[1] + h*f( X[1], Y[1] );
```

$$Y_2 := 4.425432917$$

```
> X[3] := X[2] + h;
```

$$X_3 := 1.3$$

```
> Y[3] := Y[2] + h*f( X[2], Y[2] );
```

$$Y_3 := 4.677873472$$

```
> X[4] := X[3] + h;
```

$$X_4 := 1.4$$

```
> Y[4] := Y[3] + h*f( X[3], Y[3] );
```

$$Y_4 := 4.959042570$$

```
> X[5] := X[4] + h;
```

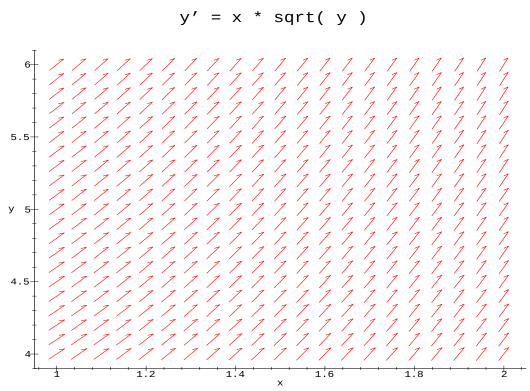
$$X_5 := 1.5$$

```
> Y[5] := Y[4] + h*f( X[4], Y[4] );
```

$$Y_5 := 5.270807280$$

To understand what has just been computed, let's look at some plots. First, create the direction field for this problem

```
> dir_plot := DEplot( ODE, [x,y], x=1..2, y=4..6, arrows=THIN ):
> display( dir_plot, title=`y' = x * sqrt( y )` );
```



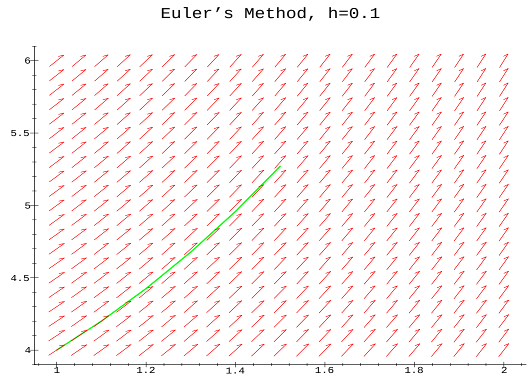
The points computed in Euler's Method can also be plotted

```
> euler_pts := [X[0], Y[0]], [X[1], Y[1]], [X[2], Y[2]],
>              [X[3], Y[3]], [X[4], Y[4]], [X[5], Y[5]];
```

$$euler\_pts := [1, 4], [1.1, 4.2], [1.2, 4.425432917], [1.3, 4.677873472],$$
$$[1.4, 4.959042570], [1.5, 5.270807280]$$

```
> euler_plot := plot( [ euler_pts ], color=GREEN, thickness=3 ):
```

```
> display( { dir_plot, euler_plot }, title=`Euler's Method, h=0.1` );
```



Note that the two plot commands (`DEplot` and `plot`) are both terminated with a colon (`:`); this prevents the displaying of the plot. The plots are displayed using the `display` command.

Euler's Method is very repetitive. In practice it would not be uncommon to use several dozen, if not hundreds, of steps. This seems natural for writing as a loop. Here is how this can be done in Maple.

```
> euler_pts := [ X[0], Y[0] ]:
> for n from 0 to N-1 do
>   X[n+1] := X[n] + h;
>   Y[n+1] := Y[n] + h * f( X[n], Y[n] );
>   new_pt := [ X[n+1], Y[n+1] ];
>   euler_pts := euler_pts, new_pt;
> od:
> print( euler_pts );
```

$$[1,4],[1.1,4.2],[1.2,4.425432917],[1.3,4.677873472],[1.4,4.959042570],[1.5,5.270807280]$$

Obtain a new approximation for the solution at $x = 1.5$ using a) twice as many steps and b) ten times as many steps as the original problem.

a)

```
> h := ??;
> N := ??;
```

Now, copy the loop that computes Euler's Method to the input region immediately following this text. Then, execute the commands in the loop.

```
> Euler( ??, ??, ??, ?? );
```

Your output should look like:

$$[1,4],[1.05,4.10],[1.10,4.206304398],[1.15,4.319105421],\\ [1.20,4.438604552],[1.25,4.565012528],[1.30,4.698549340],\\ [1.35,4.839444233],[1.40,4.987935707],[1.45,5.144271515],\\ [1.50,5.308708668]$$

b) (now you are on your own)

It quickly becomes tiresome to copy the commands for Euler's Method every time they are needed. It's also a little restrictive to insist that the independent variable be x and the dependent variable y. These problems can be avoided by putting the loop into a Maple procedure. Here is how that would look:

```
> Euler := proc( f, ic, h, N )
>   local n, euler_pts, new_pt, X, Y;
>   X[0] := ic[1];
>   Y[0] := ic[2];
>   euler_pts := [ X[0], Y[0] ]:
>   for n from 0 to N-1 do
>     X[n+1] := X[n] + h;
>     Y[n+1] := Y[n] + h * f( X[n], Y[n] );
>     new_pt := [ X[n+1], Y[n+1] ];
>     euler_pts := euler_pts, new_pt;
>   od:
>   RETURN( euler_pts );
> end;

Euler := proc(f,ic,h,N)
         local n,euler_pts,new_pt,X,Y;
             X[0] := ic[1];
             Y[0] := ic[2];
             euler_pts := [X[0],Y[0]];
             for n from 0 to N-1 do
                 X[n+1] := X[n]+h;
                 Y[n+1] := Y[n]+h*f(X[n],Y[n]);
                 new_pt := [X[n+1],Y[n+1]];
                 euler_pts := euler_pts,new_pt
             od;
             RETURN(euler_pts)
         end
```

To illustrate the use of `Euler`, let's recompute the solution to the original problem.

```
> Euler( f, [1,4], 0.1, 5 );
```

$$[1,4],[1.1,4.2],[1.2,4.425432917],[1.3,4.677873472],[1.4,4.959042570],\\ [1.5,5.270807280]$$

The examples considered so far have been rather simple. Just to illustrate the use of Euler's Method in a slightly more complicated setting, consider Problem # 16 (p. 29).

```
> ODE := diff( T(t), t ) = K*(M^4-T^4);
```

$$ODE := \frac{\partial}{\partial t}\,\mathrm{T}(\,t\,) = K\,(\,M^4 - T^4\,)$$

```
> K := 40^(-4);
```

$$K := \frac{1}{2560000}$$

```
> M := 70;
```

$$M := 70$$

This problem, which is an application of Stefan's Law of Radiation, can be solved by creating a function that evaluates to the right-hand side of the differential equation (with appropriate values for the constants) and a single call to `Euler`.

```
> f := (t,T) -> K*(M^4-T^4);
```

$$f := (\,t, T\,) \rightarrow K\,(\,M^4 - T^4\,)$$

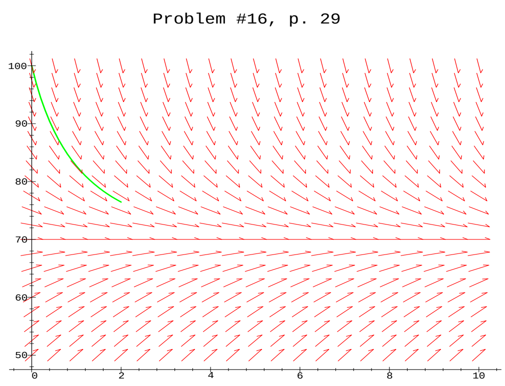
```
> temp_pts := Euler( f, [0,100], 0.1, 20 );
```

$$\begin{aligned}temp\_pts := &[\,0, 100\,], [\,.1, 97.03164063\,], [\,.2, 94.50684189\,], [\,.3, 92.32862067\,],\\ &[\,.4, 90.42790328\,], [\,.5, 88.75381371\,], [\,.6, 87.26784100\,], [\,.7, 85.94016436\,],\\ &[\,.8, 84.74724485\,], [\,.9, 83.67019423\,], [\,1.0, 82.69363876\,],\\ &[\,1.1, 81.80490839\,], [\,1.2, 80.99344589\,], [\,1.3, 80.25036815\,],\\ &[\,1.4, 79.56813510\,], [\,1.5, 78.94029616\,], [\,1.6, 78.36129346\,],\\ &[\,1.7, 77.82630729\,], [\,1.8, 77.33113343\,], [\,1.9, 76.87208475\,],\\ &[\,2.0, 76.44591161\,]\end{aligned}$$

Problem Extension:

1. Display the approximate solution computed by Euler's method on top of the direction field for this problem.

```
> dir_plot := DEplot( ODE, [t,T], t=0..10, T=50..100, arrows=THIN ):
> temp_plot := plot( [ temp_pts ], color=GREEN, thickness=3 ):
> display( { dir_plot, temp_plot }, title=`Problem #16, p. 29`  );
```



2. Repeat the Problem to determine the temperature of the body at the end of each of the first 10 minutes using a stepsize of a) h=0.1 and b) h=0.5. Plot both sets of points on top of the direction field. Which solution is better? How can you tell?

```
> temp_pts1 := Euler( ??, ??, ??, ?? );
> temp_pts2 := Euler( ??, ??, ??, ?? );
> display( { dir_plot, temp_plot1, temp_plot2 },
>          title=`Problem #16, p. 29 -- Extended` );
```

3. What can you say about the temperature in the body as t continues to increase?