

Euler's Method in Maple MA 341S, Spring, 2002

First here is the Euler script for the initial value problem $y' = y$, $y(0) = 1$, with $h = 0.1$ and 10 steps to approximate $y(1)$, which should be 2.718281828.

```
> x:=0;
> y:=1;
> deltax:=0.1;
> for i to 10
> do
> slope:= y;
> x := x + deltax;
> y := y + deltax*slope;
> end do;
```

```
      x := 0
      y := 1
deltax := .1
slope := 1
      x := .1
      y := 1.1
slope := 1.1
      x := .2
      y := 1.21
slope := 1.21
      x := .3
      y := 1.331
slope := 1.331
      x := .4
      y := 1.4641
slope := 1.4641
      x := .5
      y := 1.61051
slope := 1.61051
      x := .6
      y := 1.771561
slope := 1.771561
      x := .7
      y := 1.9487171
slope := 1.9487171
      x := .8
      y := 2.14358881
```

```

slope := 2.14358881
x := .9
y := 2.357947691
slope := 2.357947691
x := 1.0
y := 2.593742460

```

Now let's write a Maple procedure to calculate with. The procedure will have as input `x_start`, `x_end`, `y_start`, `step_size`, `the_function`.

The value of a Maple procedure is the value of the last command executed inside the procedure. For this reason I will include something extra here, namely a list of the points calculated and call the list `newpoint`. The value of `newpoint[0]` will be the starting value, and it is set in the 7th line of the procedure. The last line just before `end proc` is a loop that prints out the values of `newpoint`. Since its the last thing executed when the procedure is called, it will be the value of the procedure.

```

> euler_method:= proc(x_start,x_end,y_start,step_size,the_function)
> local N, x, y, f, j, slope, newpoint, k;
> N:=(x_end - x_start)/step_size;
> x:= x_start;
> y:= y_start;
> f:= the_function;
> newpoint[0]:= (x,y);
> for j to N
> do
> slope :=f(x,y);
> x := x + deltax;
> y := y + deltax*slope;
> newpoint[j] := (x,y);
> end do;
> for k from 0 to N
> do
> print(newpoint[k]);
> end do;
> end proc;

```

```

euler_method := proc(x_start, x_end, y_start, step_size, the_function)
local N, x, y, f, j, slope, newpoint, k;
  N := (x_end - x_start)/step_size;
  x := x_start;
  y := y_start;
  f := the_function;
  newpoint_0 := x, y;
  for j to N do
    slope := f(x, y); x := x + delta_x; y := y + delta_x * slope; newpoint_j := x, y
  end do;
  for k from 0 to N do print(newpoint_k) end do
end proc

> euler_method(0,1,1,0.1, (a,b)-> b);
      0, 1
      .1, 1.1
      .2, 1.21
      .3, 1.331
      .4, 1.4641
      .5, 1.61051
      .6, 1.771561
      .7, 1.9487171
      .8, 2.14358881
      .9, 2.357947691
     1.0, 2.593742460

```