

Euler Method for First Order ODE

Date: Jan 21, 2002

Last Revision: Jan 21, 2002

Maple 6

Bent E. Petersen

bent@alum.mit.edu

petersen@math.orst.edu

Course: Mth 256

Term: Winter 2002

File name: 256w2002-euler-method.mws

Maple has a number of numeric solvers for differential equations built-in. Normally Maple chooses a numeric method when we request a numeric answer, but we can specify the method. In particular we can specify the simple Euler method. Our goal here though is to study the Euler method and therefore we roll our own Euler method.

```
> restart;
```

Here is a simple Euler procedure for solving $dy/dx=f(x,y)$, $y(x_0)=y_0$ on a specified interval with a specified number of steps. We do no error checking - be careful. We return the approximate solution as a list of lists - the first is the list of abscissae, the second is the list of ordinates. This form is convenient for plotting.

First here is our Euler routine, eul(). Note we force Maple to use floating point arithmetic. Otherwise Maple will attempt exact evaluation, which will waste immense amounts of time and RAM.

```
> eul:=proc (f, x0, y0, x1, N)
>   local h, XX, YY, k, x, y;
>   x:=evalf(x0): XX:=[x];
>   y:=evalf(y0): YY:=[y];
>   h:=(x1-x0)/N;
>   for k from 1 to N do
>     y:=y+h*f(x, y);
>     x:=x+h;
>     XX:=[op(XX), x];
>     YY:=[op(YY), y];
>   od;
>   [XX, YY];
> end;
```

Here is a routine to convert the data returned by `eul()` into a piecewise linear (or polygonal) expression:

```
> eulpoly:=proc (EUL, x);  
>   spline(EUL[1], EUL[2], x, 1);  
> end;
```

Let's try an example. Let

```
> f01 := (x, y) -> x^2 + y^2;
```

$$f01 := (x, y) \rightarrow x^2 + y^2$$

```
> init01 := y(0) = 0;
```

$$init01 := y(0) = 0$$

```
> ode01 := diff(y(x), x) = f01(x, y(x));
```

$$ode01 := \frac{\partial}{\partial x} y(x) = x^2 + y(x)^2$$

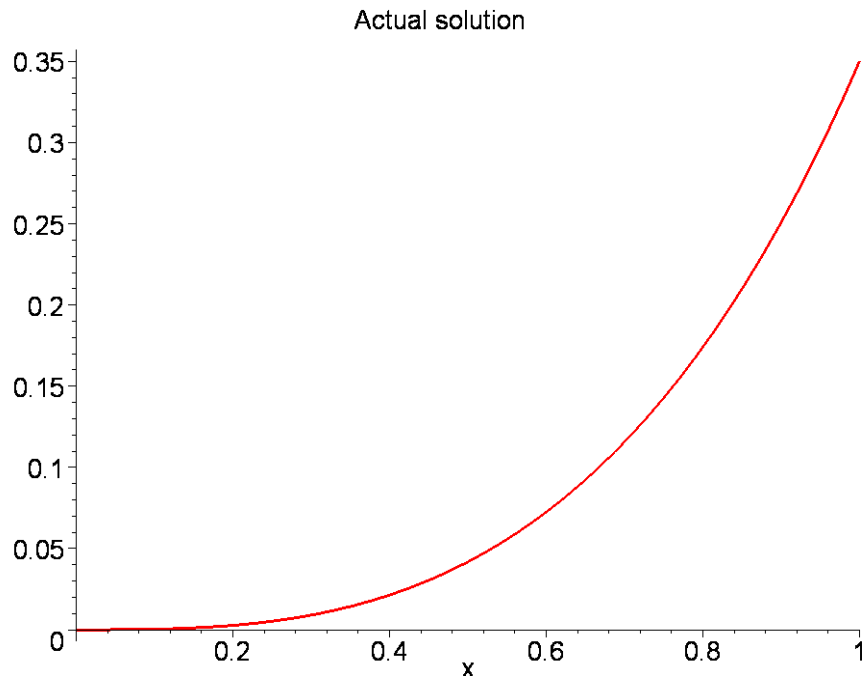
Maple can solve this initial value problem.

```
> soln01 := dsolve({ode01, init01}, y(x));
```

$$soln01 := y(x) = -\frac{x \left(-\text{BesselJ}\left(\frac{-3}{4}, \frac{1}{2}x^2\right) + \text{BesselY}\left(\frac{-3}{4}, \frac{1}{2}x^2\right) \right)}{-\text{BesselJ}\left(\frac{1}{4}, \frac{1}{2}x^2\right) + \text{BesselY}\left(\frac{1}{4}, \frac{1}{2}x^2\right)}$$

Let's plot this solution on $[0,1]$.

```
> plot(rhs(soln01), x=0..1, title="Actual solution", thickness=3);
```



Now let's see how well our Euler routine does.

```
> esoln01:=eulpoly(eul(f01,0,0,1,20),x);
```

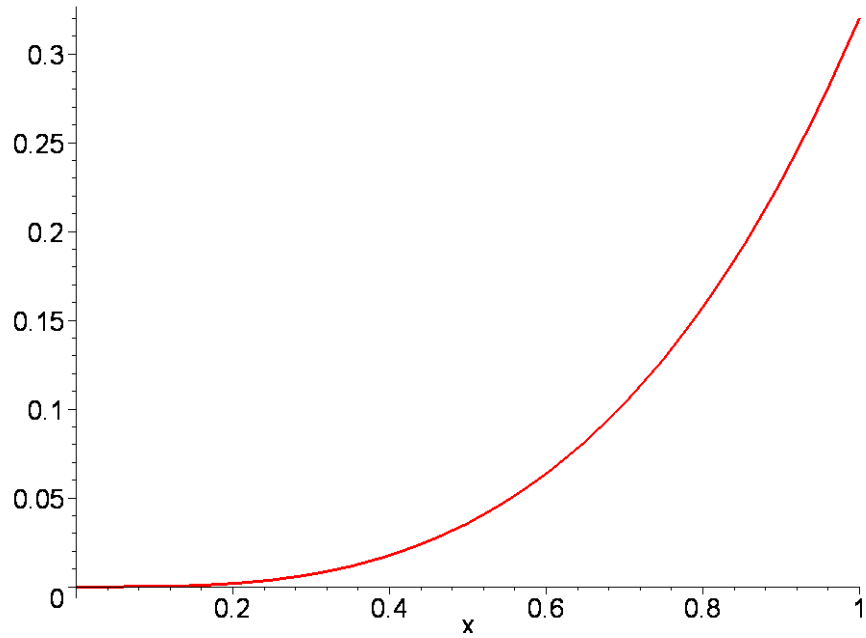
```

                                0.                                x < .05000000000
-0.0001250000000 + .002500000000 x          x < .10000000000
-0.0008750015620 + .01000001562 x          x < .15000000000
-0.002750057815 + .02250039064 x          x < .20000000000
-0.006250592199 + .04000306256 x          x < .25000000000
-0.01187834251 + .06251406380 x          x < .30000000000
-0.02013830670 + .09004727778 x          x < .35000000000
-0.03154207202 + .1226294644 x           x < .40000000000
-0.04661292226 + .1603065900 x           x < .45000000000
-0.06589314427 + .2031515278 x           x < .50000000000
-0.08995400497 + .2512732492 x           x < .55000000000
-0.1194089549 + .3048277036 x            x < .60000000000
-0.1549307431 + .3640306840 x            x < .65000000000
-0.1972733308 + .4291731266 x            x < .70000000000
-0.2472997782 + .5006394800 x            x < .75000000000
-0.3060177207 + .5789300700 x            x < .80000000000
-0.3746246135 + .6646886860 x            x < .85000000000
-0.4545658691 + .7587372220 x            x < .90000000000
-0.5476101965 + .8621198080 x            x < .95000000000
-0.6559483086 + .9761599260 x            otherwise

```

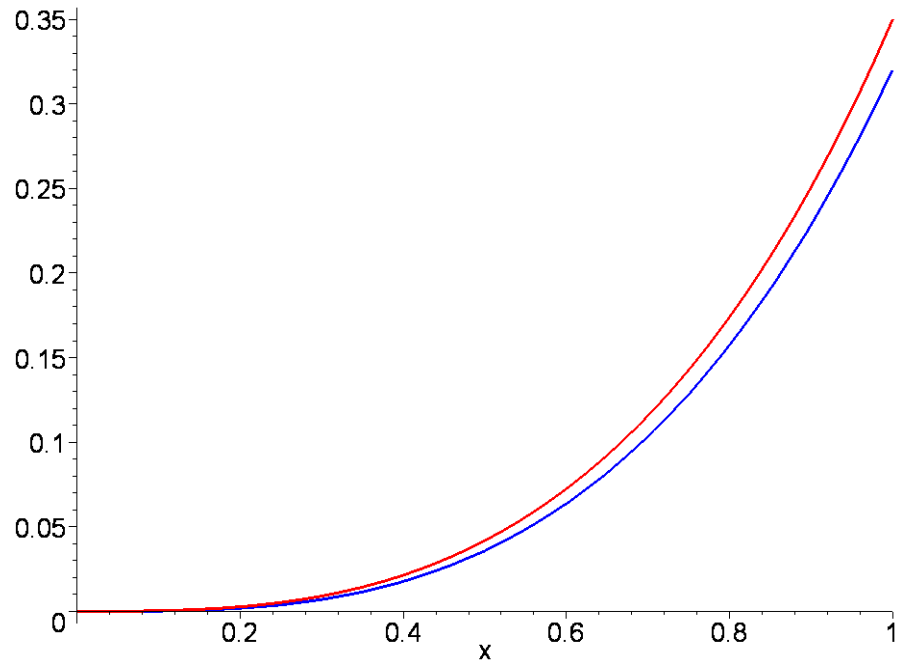
```
> plot(esoln01,x=0..1,title="Euler approximation",thickness=3);
```

Euler approximation



Let's put both plots on one diagram.

```
> plot([rhs(soln01), esoln01], x=0..1, thickness=3, color=[red, blue]);
```



Here's another example:

```
> f02 := (x, y) -> x^2 - x^3 * cos(y);
```

$$f02 := (x, y) \rightarrow x^2 - x^3 \cos(y)$$

```
> ode02 := diff(y(x), x) = f02(x, y(x));
```

$$ode02 := \frac{\partial}{\partial x} y(x) = x^2 - x^3 \cos(y(x))$$

```
> init02:=y(0)=1;
```

```
init02 := y(0) = 1
```

```
> soln02:=dsolve({ode02,init02},y(x));
```

```
soln02 :=
```

Maple returns an empty solution (after a long time) to indicate that it can not find an analytic solution expression. We can still, however, ask Maple for a numeric (approximate) solution:

```
> soln02:=dsolve({ode02,init02},y(x),numeric, output=listprocedure);
```

```
soln02 := [x = (proc(x) ... end proc), y(x) = (proc(x) ... end proc)]
```

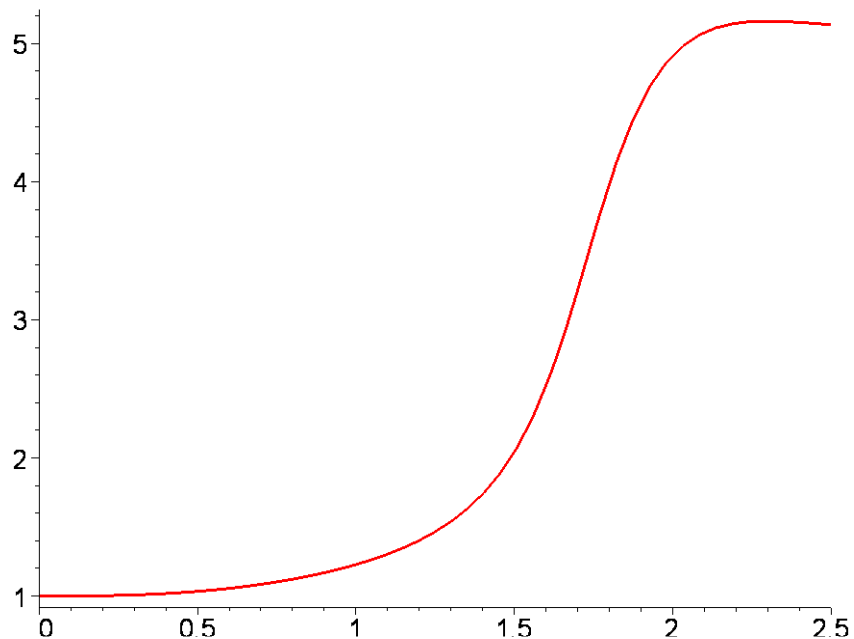
Maple actually returns a procedure to compute the numeric estimate. We can use `subs()` to separate out the function we are looking for:

```
> yy:=subs(soln02,y(x));
```

```
yy := proc(x) ... end proc
```

```
> plot(yy,0..2.5,thickness=3,title="Maple numeric solution");
```

```
Maple numeric solution
```



A better way to plot Maple's numeric solutions of ODEs is to use the `odeplot()` function. Check Maple's help.

Let's see how our Euler routine fares here

```
> esoln02:=eulpoly(eul(f02,0,1,2.5,30),x);
```

```

1. x < .08333333333
.9994473530 + .006631763994 x x < .1666666667
.9963395590 + .02527852800 x x < .2500000000
.9891359980 + .05409277200 x x < .3333333333
.9767256580 + .09132379200 x x < .4166666666
.9583478730 + .1354304760 x x < .4999999999
.9334499850 + .1852262520 x x < .5833333332
.9014682660 + .2400520560 x x < .6666666665
.8615149061 + .2999820960 x x < .7499999998
.8119402961 + .3660815760 x x < .8333333331
.7497114661 + .4407561720 x x < .9166666664
.6694917112 + .5282686320 x x < .9999999997
.5622046300 + .6355557128 x x < 1.083333333
.4126790717 + .7735793051 x x < 1.166666666
.1956310591 + .9596204580 x x < 1.249999999
-.1313618563 + 1.221214791 x x < 1.333333332
-.6401867843 + 1.602833487 x x < 1.416666665
-1.452986322 + 2.176574337 x x < 1.499999998
-2.771798800 + 3.055782657 x x < 1.583333331
-4.892291940 + 4.395041484 x x < 1.666666664
-8.054363707 + 6.292284547 x x < 1.749999997
-11.62204080 + 8.330957175 x x < 1.833333330
-12.37127935 + 8.739632744 x x < 1.916666663
-7.069027898 + 5.973240675 x x < 1.999999996
-4.934920761 + 2.685472758 x x < 2.083333329
3.201016425 + .912108675 x x < 2.166666662
4.885510847 + .134649709 x x < 2.249999995
5.543624498 - .157845247 x x < 2.333333328
5.708878901 - .228668563 x x < 2.416666661
5.690010983 - .220861149 x otherwise

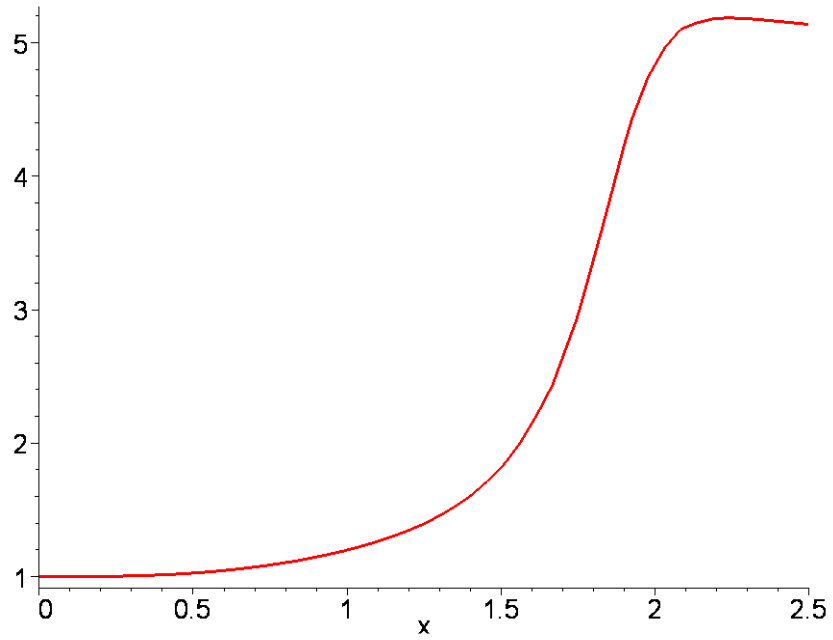
```

```

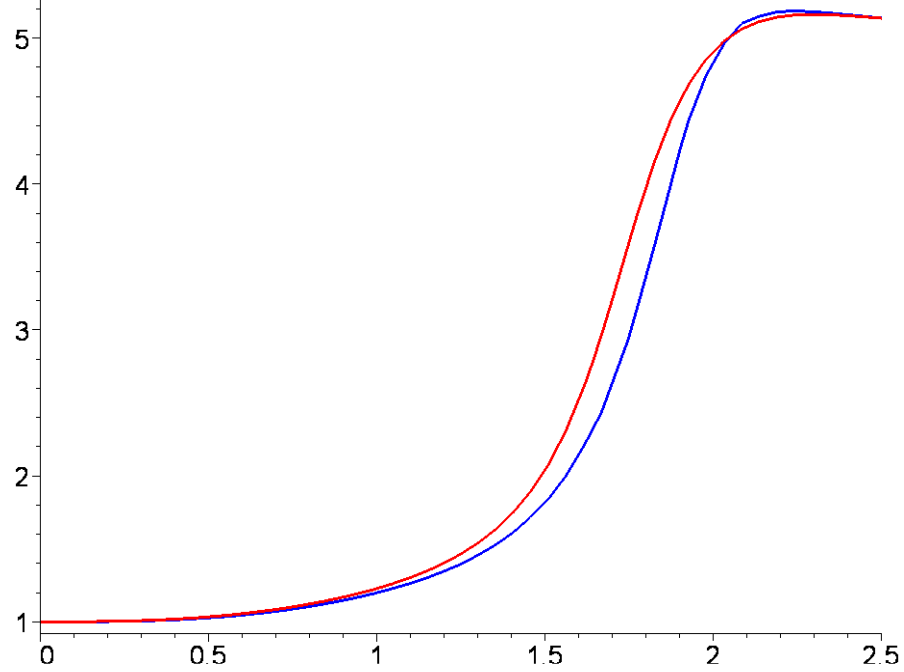
> plot(esoln02,x=0..2.5,title="Euler approximation",thickness=3);

```

Euler approximation



```
> plot ( [yy,  
unapply(esoln02,x)], 0..2.5, thickness=3, color=[red,blue] );
```



Not bad, considering how rough the Euler method is.

```
>
```