\>

prob1-4-2.mws

Solution to Problem # 2 in Section 1.4 of BDH
(worksheet based upon template.mws)

\>

Getting started

Solution

The problem involves the following information

```
> MODEL := diff( y(t) , t ) = t - y(t)^2 ; # define the differential equation model
> IC := y(0) = 1;                          # specify the initial condition
>
```

Preliminary Analysis

Numerical Solution using Euler's Method

Now we get to the numerical solution. To begin, copy the definition of the `Euler` command from template.mws.

DO NOT CHANGE ANYTHING IN THE FOLLOWING INPUT REGION - YOU HAVE BEEN WARNED !!!

```
> Euler := proc( MODEL, IC, DOMAIN, N )
>   local a, b, dt, dy, f, i, tt, yy, LISTpts, vars;
>   vars := (op(2,lhs(MODEL) ), op([1,0],lhs(MODEL) ) );
>   f := unapply( rhs(MODEL), vars );
>   a := op(1,rhs(DOMAIN)); b := op(2,rhs(DOMAIN));
>   dt := (b-a)/N;
>   tt := a;
>   yy := rhs(IC);
>   LISTpts := [ tt, yy ];
>   for i from 1 to N do
>     dy := evalf( f(tt,yy) ) * dt;
>     yy := yy + dy;
>     tt := tt + dt;
>     LISTpts := LISTpts, [ tt, yy ];
>   od; RETURN( [ LISTpts ] );
> end:
>
```

Let's continue our preparations by checking that all necessary assignments have been made.

```
> MODEL;
```

$$\frac{\partial}{\partial t} y(t) = t - y(t)^2$$

```
> IC;
```

$$y(0) = 1$$

```
> VAR;
```

$$\{y(t)\}$$

```
> DOMAIN ;
```

$$t = 0 .. 1$$

\>

The last ingredient is the number of steps to be used in approximating the solution on the above domain. In this case, dt = 0.25 so that 4 steps are needed to span the interval 0 .. 1.

```
> Nsteps := 4;
```

$$Nsteps := 4$$

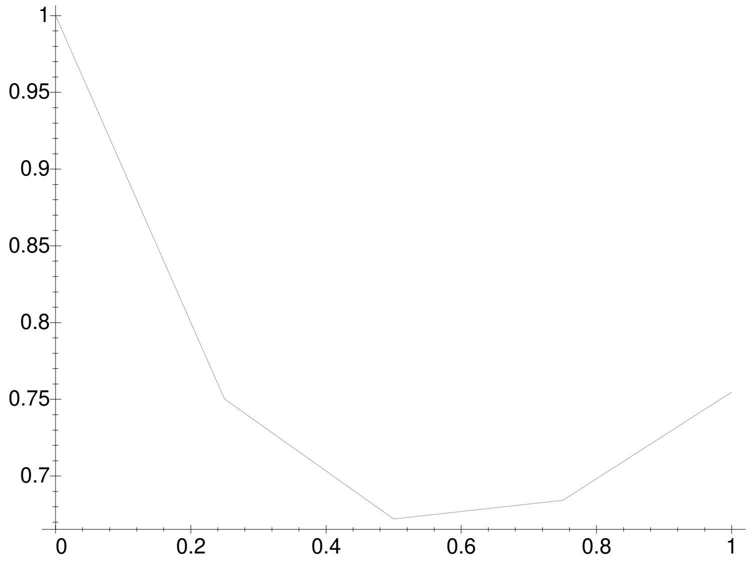
\>

The points, and corresponding plot, computed by Euler's method are:

```
> ptsE := Euler( MODEL, IC, DOMAIN, Nsteps );
```

$$ptsE := \left[[0, 1], \left[\frac{1}{4}, .7500000000\right], \left[\frac{1}{2}, .6718750000\right], \left[\frac{3}{4}, .6840209961\right], [1, .7545498153] \right]$$

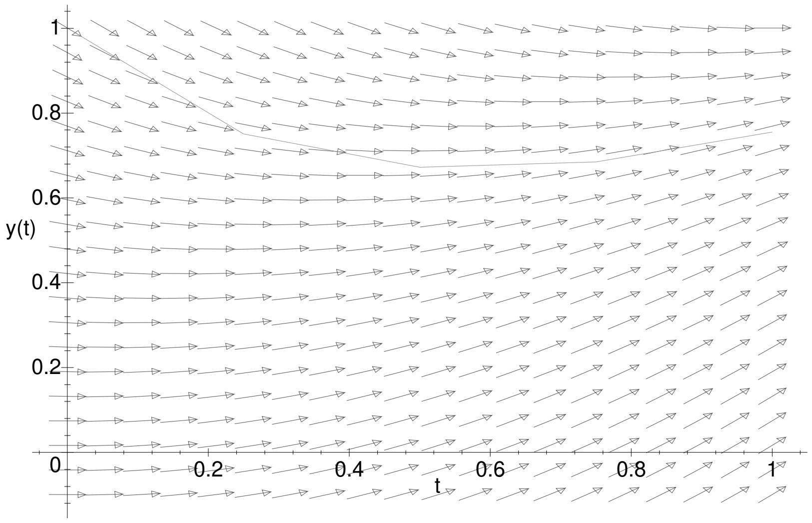
```
> plotE := plot( ptsE, color = GREEN, thickness = 2 ):
> plotE;
```




```
>
```

Note that this plot uses a different vertical scale than the earlier plots. We could modify the `plot` command, but a better idea is to overlay the approximate solution on the slope field and see how well the solution follows the slope field.

```
> display( [ plotSLOPE, plotE ] );
```



Observe that the approximate solution starts out following the slope field but starts to cross the slope field towards the end of each segment of the plotted solution.

```
>
> dt := (1-0)/Nsteps;
>
> solnE:= dsolve( { MODEL, IC }, VAR, numeric, method = classical[foreuler], stepsize =
  dt);
>
> NUMpts := 4;
> odeplot( solnE, [t, y(t)], rhs(DOMAIN), style = line, color = BLUE, numpoints =
  NUMpts );
>
>
> restart;
```