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

Numerical Methods for First-Order IVPs
A Maple Worksheet to Accompany Section 3.5 of Nagel and Saff

Math 242 -- Fall 1994

Doug Meade (meade)
17 February 1995

Filename: sec3-5.ms

Before we learned any of the solution techniques for ODEs, we learned about Euler's method. While Euler's method is simple to use (and program), we still need to ask a few questions. How good are the results from Euler's method? Are there better (more accurate) methods that could be used?

The fundamental concept that is needed to evaluate a numerical method and compare it with other methods, is that of ERROR. There are many different types of error that can be examined; two examples are absolute and relative errors. When the exact solution is known, it's a simple matter to compute the errors. But, in a real problem you are not likely to know the exact solution. Now what can you do?

The whole question of errors is the basis of any course in numerical analysis. We do not have the time to examine the entire error analysis process.

After you have completed this worksheet, please take a little time to think about the following questions.

1. What are the relative strengths and weaknesses of the two Euler methods?
2. When would be more likely to use the Euler methods?
3. Can you explain, in words, the ideas behind the two numerical methods?
4. Can you explain, in words, the differences between the two numerical methods?

All assigned problems in section 3.5 can be answered by making simple modifications of this worksheet.

EXAMPLE #1

Let's begin with the example used in Section 1.3 (p. 17):

y' = x *sqrt(y), y(1)=4

```
> f := (x,y) -> x*sqrt(y);
  x[0] := 1:
  y[0] := 4:
  h := 0.1:
  N := 5:
```

Here's the Maple implementation of Euler's method.

```
> 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;
    print( new_pt );
  od:
  euler_plot := plot( [euler_pts],
                      style=POINT, color=RED, symbol=DIAMOND ):
>
```

The IMPROVED Euler method is only a minor modification of Euler's method. The basic idea is to replace the slope at the initial point with the average of the slopes at the initial point and a simple guess of the terminal point. Euler's method is used to determine the simple guess of the terminal point.

```
> imp_euler_pts := [ x[0],  y[0] ]:
  for n from 0 to N-1 do
    F := f(x[n],y[n]);
    G := f(x[n]+h,y[n]+h*F);
    x[n+1] := x[n] + h;
    y[n+1] := y[n] + h/2*(F+G);
    new_pt := [ x[n+1], y[n+1] ];
    imp_euler_pts := imp_euler_pts, new_pt;
    print( new_pt );
  od:
  imp_euler_plot := plot( [imp_euler_pts],
                          style=POINT, color=GREEN, symbol=BOX ):
>
```

For this problem it is possible to find an exact solution (as a separable ODE).

```
> exact_soln := rhs( dsolve( {diff(Y(x),x)=f(x,Y), Y(x[0])=y[0]}, Y(x) ) );
> exact_plot := plot( exact_soln, x=1..1.5 ):
```

It is now possible to compare the exact solution with the solutions computed by Euler's method and improved Euler's method. Notice how much more accurate the solution obtained by the Improved Euler's Method is!

```
> display( {euler_plot,imp_euler_plot,exact_plot},
           title='Exact, Euler, and improved Euler solutions');
```

To really appreciate the differences between the two numerical methods, let's examine the errors that result from the two different algorithms. In particular, let's concentrate on the relationship between the error and the stepsize.

EXAMPLE #2

Consider the first-order linear IVP

y' = sin(t) - 1/2 y, y(0)=0

Use Euler's method to approximate the value of the solution at t=4 with a tolerance of no more than epsilon=0.01. (Use the absolute error for your stopping criteria.)

Hint: To organize your results construct a table with 4 columns: h, y(4), absolute error, relative error

```
> f := (x,y) -> sin(x) - y/2;
  x[0] := 0:
  y[0] := 0:
  h := 0.125;
  N := 32;
```

We begin with regular Euler's method

```
> 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;
    print( new_pt );
  od:
>
```

Then, repeat the same problem using improved Euler's method

```
> f := (x,y) -> sin(x) - y/2;
  x[0] := 0:
  y[0] := 0:
  h := 0.125 ;
  N := 32 ;
> imp_euler_pts := [ x[0],  y[0] ]:
  for n from 0 to N-1 do
    F := f(x[n],y[n]);
    G := f(x[n]+h,y[n]+h*F);
    x[n+1] := x[n] + h;
    y[n+1] := y[n] + h/2*(F+G);
    new_pt := [ x[n+1], y[n+1] ];
    imp_euler_pts := imp_euler_pts, new_pt;
    print( new_pt );
  od:
>
```

Compare the results of the two experiments. Look at the absolute errors - does these errors decrease faster with one of the methods? What about the relative errors? What does this say about the two different numrical methods?

END OF WORKSHEET

```
>
```