9.1 Better numerical solutions than Euler's

a lesson for MATH F302 Differential Equations

Ed Bueler, Dept. of Mathematics and Statistics, UAF

March 8, 2019

for textbook: D. Zill, *A First Course in Differential Equations with Modeling Applications*, 11th ed.

the situation

these three facts make solving differential equations interesting:

1. DEs from science and engineering are usually nonlinear
2. the by-hand methods which dominate Math 302[1] are all weak
   - mostly they apply to linear DEs
   - often they need the linear DE to have special coefficients
     - "special" = constant, analytic, ...
3. on the other hand, Euler's method is too inaccurate

$$y_{n+1} = y_n + h\,f(t_n, y_n) \tag{1}$$

   - "*whatever advantage* (1) *has in its simplicity is lost in the crudeness of its approximations.*" Zill, p. 369
   - *but*, at least Euler's method does not care if your DE is linear

[1] Chapters 2,4,6,7,8

can we do better than Euler?

- here is the basic DE:

$$\frac{dy}{dt} = f(t, y)$$

  - it could be a single equation or a system (§4.10,5.3)
  - in §9.1 and 9.2 we stick to single first-order DEs
- good thing: numerical methods do not care if a DE is linear!
- so we start again with Euler's method

$$y_{n+1} = y_n + h\, f(t_n, y_n)$$

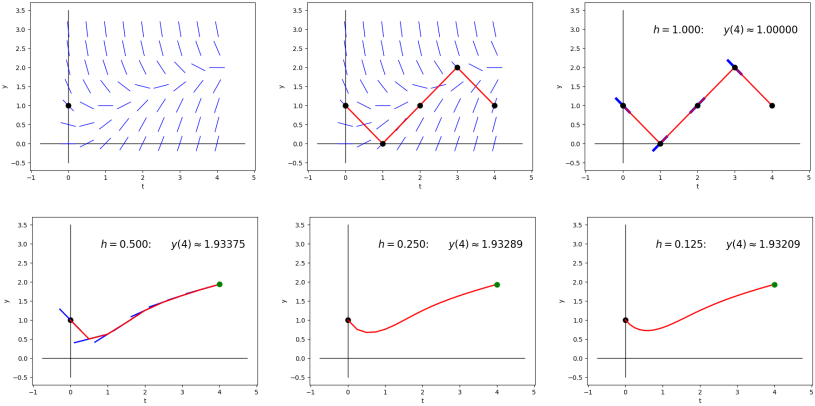
  or equivalently

$$\frac{y_{n+1} - y_n}{h} = f(t_n, y_n)$$

  and try to do better

see slides and video on Euler's method

- see my §2.6 slides and video
  - you must understand everything in those slides!
- they showed this sequence for $\frac{dy}{dt} = t - y^2$, $y(0) = 1$:



  - this visualization needs to make sense! review Euler's method!

Euler's method is a short code

```
function [t, y] = euler1(f,tspan,y0,h)
% EULER1  Euler's method for ODE IVP
%     dy/dt = f(t,y),  y(t0) = y0
% Second argument is tspan = [t0, tf].  Computes steps of size h to
% approximate y(tf).  Example:
%    >> f = @(t,y) t - y^2;
%    >> [tt,yy] = euler1(f,[0,4],1,0.5);
%    >> plot(tt,yy)
% Compare IMPROVED2, RK4, and ODE45.

M = round((tspan(2)-tspan(1))/h);   % get number of steps
t = linspace(tspan(1),tspan(2),M+1);
y = zeros(size(t));
y(1) = y0;
for n = 1:M
    y(n+1) = y(n) + h * f(t(n),y(n));
end
```

example with `euler1.m`

- you can run this with Octave Online or MATLAB or OCTAVE
- see comments:

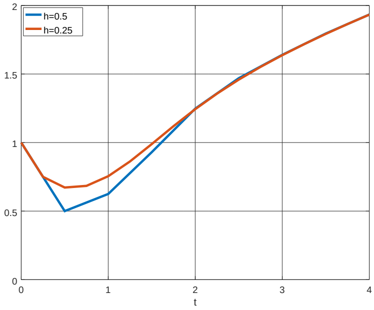
```
>> help euler1
```

- run the example:

```
>> f = @(t,y) t - y^2;
>> [tt,yy] = euler1(f,[0,4],1,0.5);
>> plot(tt,yy)
```

- reduce step size and overlay:

```
>> [tt,yy] = euler1(f,[0,4],1,0.25);
>> hold on
>> plot(tt,yy,'r')
```



`euler1.m`: explaining the lines

- the top line declares what are inputs and outputs:

```
function [t, y] = euler1(f,tspan,y0,h)
```

  - `tspan` is vector of two numbers: $[t_0, t_f] =$`[tspan(1),tspan(2)]`

- then there is a block of comments
- the first "real" line computes the number of steps wanted by user, based on h and $[t_0, t_f]$: $M = (t_f - t_0)/h$

```
M = round((tspan(2)-tspan(1))/h);
```

- given the number of steps, values t_n can be pre-computed:

```
t = linspace(tspan(1),tspan(2),M+1);
```

  - for example, `linspace(0,4,5)` is the list $[0, 1, 2, 3, 4]$
  - same as `t = tspan(1):h:tspan(2);` if user is careful

- allocate empty space for solution: `y = zeros(size(t));`
- remainder is Euler's method itself:

```
y(1) = y0;
for n = 1:M
    y(n+1) = y(n) + h * f(t(n),y(n));
end
```

from Euler to ode45 in three steps

- our use of euler1 should look familiar

```
>> f = @(t,y) t - y^2;
>> [tt,yy] = euler1(f,[0,4],1,0.5);
>> plot(tt,yy)
```

  - just like how we used ode45 in §5.3 and §4.10

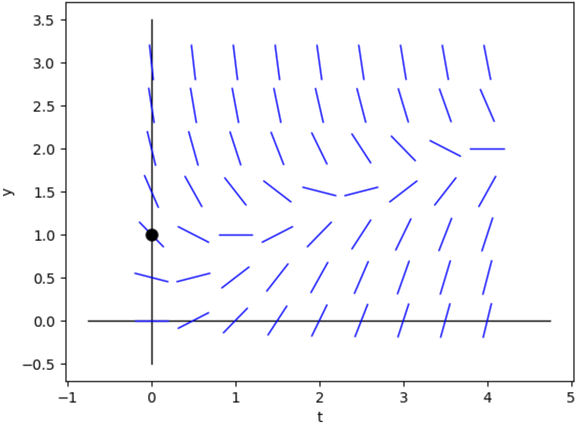
- I will show three new codes:

```
function [t,y] = euler1(f,tspan,y0,h)
function [t,y] = improved2(f,tspan,y0,h)
function [t,y] = rk4(f,tspan,y0,h)      % section 9.2
function [t,y] = ode45(f,tspan,y0)
```

  - all have the same inputs and same size outputs:
  - they approach the black box ode45 in quality
  - only difference versus ode45: it does not require choice of h

better than Euler

- start again with direction field for $y' = f(t, y) = t - y^2$
- *improved Euler* takes an Euler step of length h to a temporary value y^*, then averages slopes at the two known points, then uses the average slope for a step of length h
- visual version:



improved Euler as formula

- it takes an Euler step of length h to a temporary value y^*

$$y^* = y_n + hf(t_n, y_n)$$

- ...then averages slopes at known points

$$m_{\text{av}} = \frac{1}{2}\left(f(t_n, y_n) + f(t_{n+1}, y^*)\right)$$

- ...then uses the average slope for a step of length h

$$y_{n+1} = y_n + h\, m_{\text{av}}$$

- thus:

$$\boxed{\begin{aligned} y^* &= y_n + hf(t_n, y_n) \\ y_{n+1} &= y_n + \frac{h}{2}\left[f(t_n, y_n) + f(t_{n+1}, y^*)\right] \end{aligned}}$$

- or (ugly):

$$y_{n+1} = y_n + \frac{h}{2}\left[f(t_n, y_n) + f\left(t_{n+1}, y_n + hf(t_n, y_n)\right)\right]$$

new code just like the old code

- my new code `improved2.m` is just like `euler1.m`
- *except* inside the `for` loop:

```
function [t, y] = improved2(f,tspan,y0,h)
...
for n = 1:M
    ystar = y(n) + h * f(t(n),y(n));
    y(n+1) = y(n) + h * ( f(t(n),y(n)) + f(t(n+1),ystar) ) / 2;
end
```

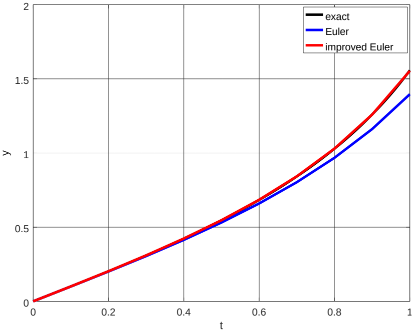
- let's use a problem where we know the exact solution
- *example.* $y' = 1 + y^2, \quad y(0) = 0$
- *exact solution.* it's separable

$$\begin{aligned} \int \frac{dy}{1+y^2} &= \int dt \\ \arctan(y) &= t + c \\ y(t) &= \tan t \end{aligned}$$

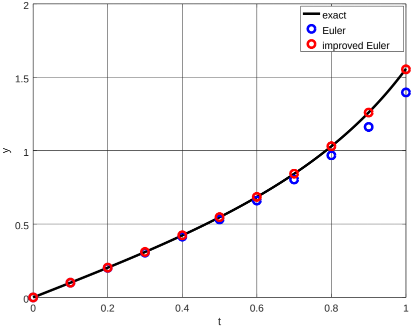
- Euler and improved Euler solutions for $y(1)$ with step $h = 0.1$:

```
>> f = @(t,y) 1+y^2;
>> [tt,ye] = euler1(f,[0,1],0,0.1);
>> [tt,yie] = improved2(f,[0,1],0,0.1);
>> ye(end), yie(end), tan(1)
ans =  1.3964
ans =  1.5538
ans =  1.5574
```

plotting style for truth and justice




```
>> plot(t,yexact,'k',...
        tt,ye,'b',tt,yie,'r')
```




```
>> plot(t,yexact,'k',...
        tt,ye,'bo',tt,yie,'ro')
```

the §9.1 WebAssign homework is easy

- get on Octave Online or MATLAB/OCTAVE
- yes, you may and should *use* the codes I have posted at the Codes tab of the course website
- use `improved2` exactly the way I did two slides back
- if you have technical difficulties then *post to Piazza*!
  - anonymous is fine, but make it a public post for efficiency

- *exercise #9.* Use the improved Euler's method to obtain a four-decimal approximation of the indicated value. First use $h = 0.1$ and then use $h = 0.05$.

$$y' = xy^2 - \frac{y}{x}, \quad y(1) = 1; \quad y(1.5)$$

solution. [fill in MATLAB/Octave code]

```
>> yy(end), yyy(end)
ans =  1.3260
ans =  1.3315
```

regarding exercise #11

- only one part of the WebAssign is not that easy ...
- how do you find the "actual value $y(0.5)$" for this ODE IVP?

$$y' = (x + y - 1)^2,\ y(0) = 2$$

- answer. we have not solved this kind of ODE before. but if you substitute $u = x + y - 1$ you can work it out

notation

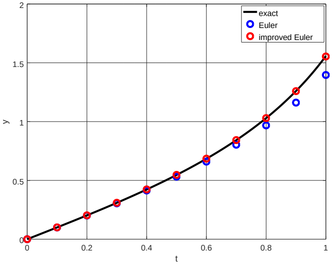
- one way to derive Euler method is using Taylor series
  ... but we need clarity about notation first
- consider $y' = f(t, y)$, as usual, with solution $y(t)$
- notation:

$$
\begin{aligned}
y(t) &= (\text{the exact solution}) \\
y(t_n) &= (\text{the exact solution evaluated at } t_n) \\
y_n &= (\text{the number computed by numerical method})
\end{aligned}
$$

- key point about notation:

  $y(t_n)$ is not the same as y_n

- absolute error $= |y_n - y(t_n)|$



order

- one may derive Euler method using Taylor series:

$$y(t+h) = y(t) + y'(t)h + \frac{y''(c)}{2}h^2$$

- or equivalently, because $t_{n+1} = t_n + h$ and $y' = f(t, y)$:

$$y(t_{n+1}) = y(t_n) + h\,f(t, y(t_n)) + \frac{y''(c)}{2}h^2$$

- drop the remainder term; use result to *define* the next value:

$$y_{n+1} = y_n + h\,f(t, y_n)$$

- Euler's method is *order 1* because we dropped the "h^2" term
- improved Euler method is *order 2* because one may derive it by dropping a "h^3" term from the Taylor series
    - not shown

improved versus *modified* Euler

- in §2.6 slides I mentioned the *modified* Euler method:

$$y^* = y_n + \frac{h}{2} f(t_n, y_n)$$
$$y_{n+1} = y_n + h\, f(t_n + \frac{h}{2}, y^*)$$

- compare *improved* Euler method (§9.1):

$$y^* = y_n + hf(t_n, y_n)$$
$$y_{n+1} = y_n + \frac{h}{2}\,[f(t_n, y_n) + f(t_{n+1}, y^*)]$$

- *exercise.* sketch each method to the right
- alternate (better) naming scheme:

name	*order*	*alternate name*
Euler	1	explicit first-order
improved Euler	2	explicit trapezoid
modified Euler	2	explicit midpoint

comment, and expectations

- it turns out that both improved Euler and modified Euler are order 2 methods from the big *Runge-Kutta* family of methods
    - §9.2 introduces an order 4 Runge-Kutta method
- just watching this video is *not* enough!
    - see "found online" videos and stuff at
      `bueler.github.io/math302/week10.html`
    - *read* section 9.1 in the textbook
    - *do* the WebAssign exercises for section 9.1