9.2 Runge-Kutta methods

a lesson for MATH F302 Differential Equations

Ed Bueler, Dept. of Mathematics and Statistics, UAF

March 16, 2019

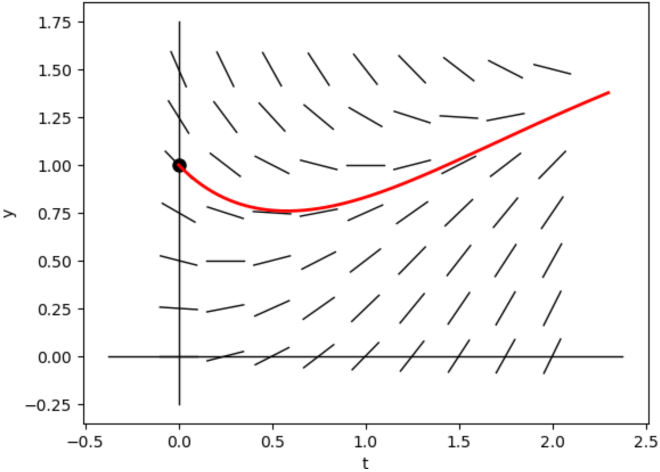
for textbook: D. Zill, *A First Course in Differential Equations with Modeling Applications*, 11th ed.

the Runge-Kutta happy family

- these slides describe and test the most-famous *Runge-Kutta* (RK) method, namely (classical) RK4 which is order 4
    - RK4 dates to $\sim$1900, *before* invention of electronic computers
    - there are dozens of useful RK methods of all orders ≥ 1
    - Euler's method is the order 1 RK method
    - improved Euler is an order 2 RK method
    - ∞ly-many methods in the family ...
- RK4 was accurate enough for most ODE solutions in science and engineering until $\sim$1960
- better computers and programming languages allow reliable/debugged implementations of better-than-RK4 methods like `ode45` in MATLAB/OCTAVE
    - ...which are *not* a lot more accurate
    - instead, modern methods like `ode45` are *adaptive* so the user does not need to choose a step size *h*

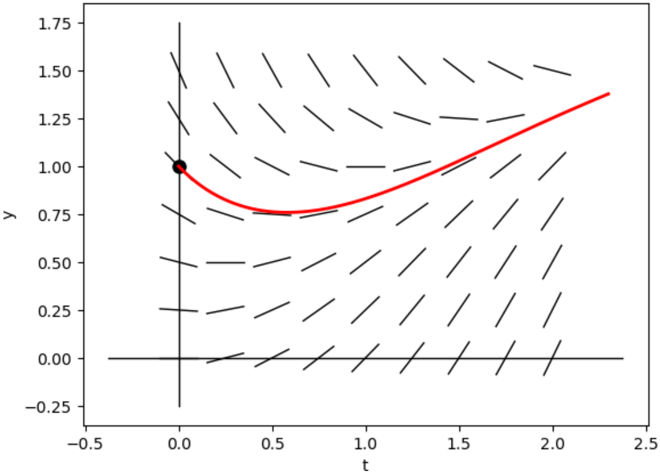
improved Euler dance

- ODE IVP: $\frac{dy}{dt} = t - y^2$, $y(0) = 1$
- show on the direction field: improved Euler with $h = 1$



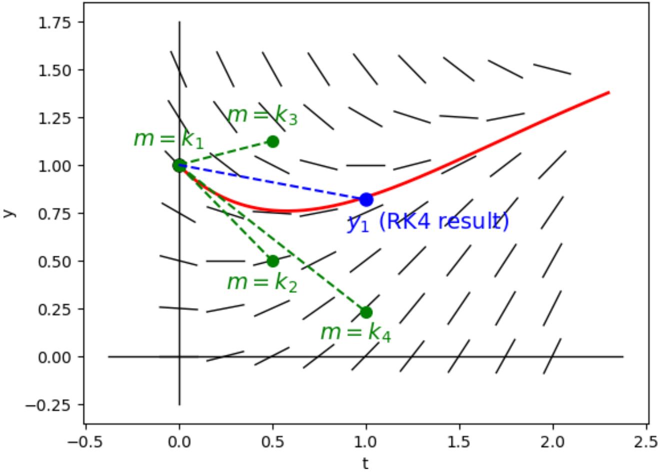
RK4 dance

- ODE IVP: $\frac{dy}{dt} = t - y^2$, $y(0) = 1$
- show on the direction field: RK4 with $h = 1$



RK4 precise dance

- ODE IVP: $\frac{dy}{dt} = t - y^2$, $y(0) = 1$
- show on the direction field: RK4 with $h = 1$



the RK4 formulas

- usually written using four slopes "k_i" from direction field
- update the y-value by a weighted average of these slopes:

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2} k_1) \\ k_3 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2} k_2) \\ k_4 &= f(t_n + h, y_n + h k_3) \end{aligned} \implies y_{n+1} = y_n + h \, \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}$$

- here are the formulas for improved Euler, written the same way:

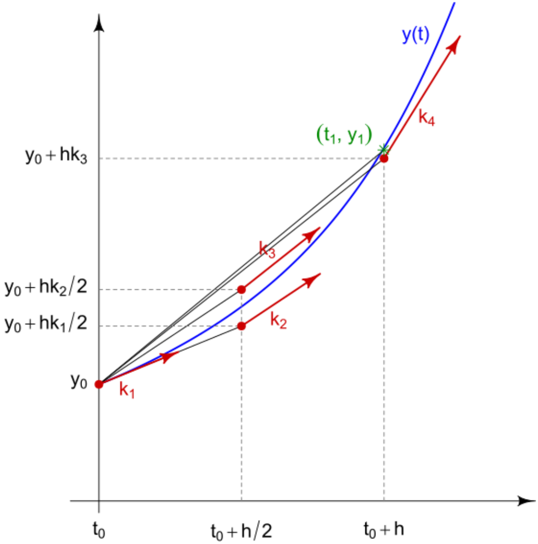
$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + h, y_n + h k_1) \end{aligned} \implies y_{n+1} = y_n + h \, \frac{k_1 + k_2}{2}$$

- Euler written the same way:

$$k_1 = f(t_n, y_n) \qquad \implies y_{n+1} = y_n + h \, k_1$$

RK4 scheme in one diagram

- drawing this sketch, or similar, will be an extra credit problem on Midterm 2



code rk4.m

- code posted at the Codes tab at the course website

```
function [t, y] = rk4(f,tspan,y0,h)
% RK4  Classical Runge-Kutta order-4 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] = rk4(f,[0,4],1,0.5);
%   >> plot(tt,yy)
% Compare EULER1, IMPROVED2, 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
    k1 = f(t(n), y(n));
    k2 = f(t(n) + h/2, y(n) + h*k1/2);
    k3 = f(t(n) + h/2, y(n) + h*k2/2);
    k4 = f(t(n) + h, y(n) + h*k3);
    y(n+1) = y(n) + (h/6) * ( k1 + 2*k2 + 2*k3 + k4 );
end
```

exercise #7 in §9.2

- *exercise.* Use the RK4 method with $h = 0.1$ to obtain a four-decimal approximation of the indicated value:

$$y' = e^{-y}, \quad y(0) = 0; \qquad y(0.5)$$

solution.

```
>> f = @(t,y) exp(-y);
>> [tt,yy] = rk4(f,[0,0.5],0,0.1);
>> yy(end)
ans =    0.40547
```

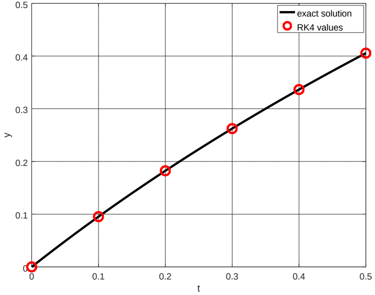
better version of same exercise

- *exercise.* Use the RK4 method with $h = 0.1$ to obtain a four-decimal approximation of the indicated value:

$$y' = e^{-y}, \quad y(0) = 0; \qquad y(0.5)$$

Compute the exact value. Plot both solutions in good style.

solution, cont.

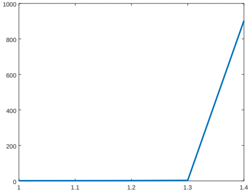



```
>> log(1+0.5)
ans =    0.40547
>> t = 0:.001:0.5;
>> plot(t,log(1+t),'k')
>> hold on,  plot(tt,yy,'ro','markersize',12)
>> xlabel t,  ylabel y,  grid on
>> legend('exact solution','RK4 values')
```

there are *bad* ODE IVPs out there!

- *exercise #15 in §9.2.* for this ODE IVP, find $y(1.4)$:

$$y' = x^2 + y^3, \quad y(1) = 1,$$



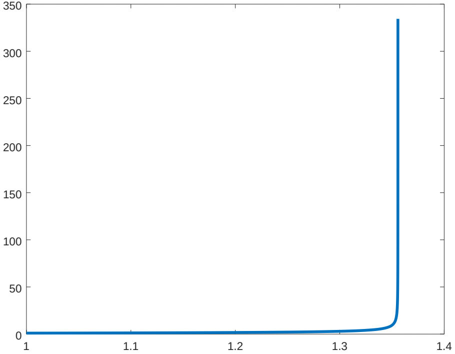
solution.

```
>> f = @(x,y) x^2 + y^3;
>> [xx,yy] = rk4(f,[1,1.4],1,0.1);
>> plot(xx,yy)
>> yy
yy =
      1      1.2511      1.6934      2.9425      903.03
>> [xxx,yyy] = ode45(f,[1,1.4],1);
warning:  Solving was not successful.  The iterative integration
loop exited at time t = 1.355695 before the endpoint at
tend = 1.400000 was reached.  This may happen if the stepsize
becomes too small.  Try to reduce the value of 'InitialStep'
and/or 'MaxStep' with the command 'odeset'.
warning: called from ...
```

when does it blow up?

- for this ODE IVP, find $y(1.4)$:[1] $\quad y' = x^2 + y^3, \quad y(1) = 1$

```
>> [xxx,yyy] = ode45(f,1:0.00001:1.35569,1);
>> plot(xxx,yyy)                                  <-- PLOT BELOW
>> [xxx,yyy] = ode45(f,1:0.00001:1.35570,1);      <-- GENERATES WARNING
```



[1] "find $y(1.4)$" *is* a trick question ... never gets that far!

RK4 and `ode45` summary

- the last example shows one reason nonlinear ODEs are interesting . . . the *problems* can be badly behaved
- but RK4 is the first of powerful tools to handle nonlinear ODE problems via highly-accurate numerical approximations
- the black-box `ode45` is a combination of certain "RK4" and "RK5" formulas
  - the two formulas use the same intermediate locations to get slopes from the direction field
  - . . . which allows adaptive step size computations
  - if you really want to know, see the Matlab technical doc page on `ode45` and the wikipedia page on the Dormand-Prince method

expectations

- just watching this video is *not* enough!
    - see "found online" videos and stuff at
      bueler.github.io/math302/week10.html
    - *read* section 9.2 in the textbook
    - *do* the WebAssign exercises for section 9.2