



11. SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS (ODE)

11.1.1 Remember: what we saw during the last lesson

Non-linear fitting using the Gauss-Newton method

Optimization searching for an extremum with-/ out using derivatives

11.1.2 Overview: what you will learn today

Euler and Runge-Kutta methods for solving ODEs

Predator-prey model using a system of ODEs

Anharmonic pendulum writing a 2nd order equation as a system with two ODEs

Warning instability, stiff problem

11.2 Euler and Runge-Kutta methods (NAM 8.2, H 9.3)

Problem: solve the ordinary differential equation for $y(t)$ for $t > t_0$

$$y' = f(t, y), \quad \text{with } y(t_0) = y_0$$

using approximations with small steps t_0, t_1, \dots such that $t_{i+1} = t_i + h$.

Euler's method can directly be derived from the forward difference

$$\frac{y(t_i + h) - y(t_i)}{h} \approx f(t, y), \quad \Rightarrow \quad \boxed{y(t_{i+1}) \approx y_i + hf(t_i, y_i)}$$

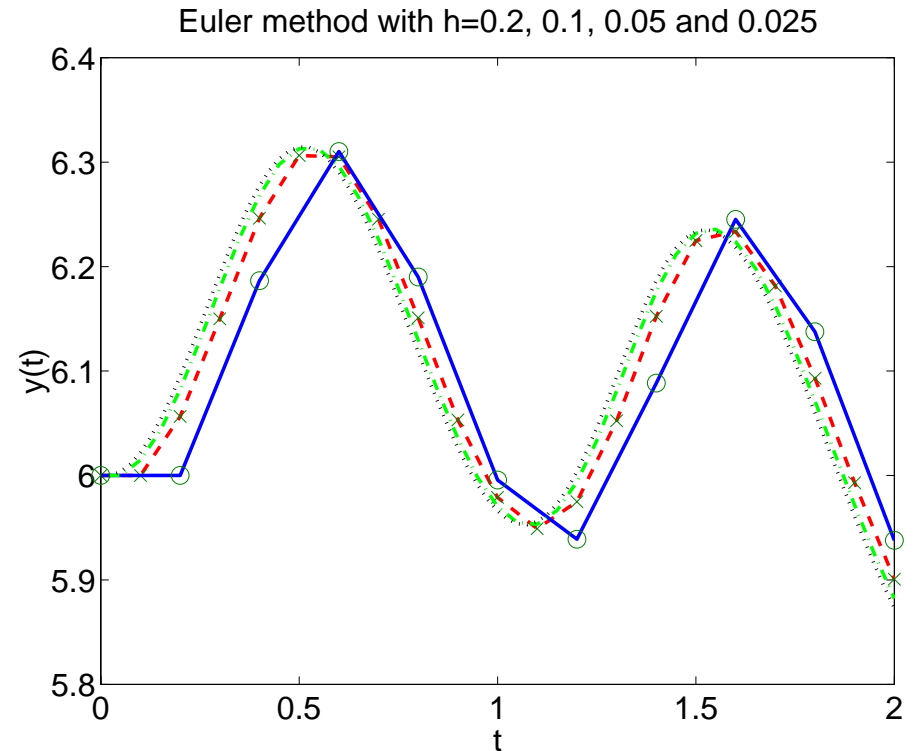
Starting from the *initial condition* (t_0, y_0) , one step with Euler's method produces an approximation (t_1, y_1) with a *local error* $\mathcal{O}(h^2)$; after n steps to reach the final time $t_n = t_0 + nh$, the solution (t_n, y_n) accumulates a *global error* $n\mathcal{O}(h^2) \sim \mathcal{O}(h)$.

Example solve $y'(t) = \sin(ty)$ with $y(0) = 6$ using Euler's method

```
function fend=feuler(h)
y=6; t=0; tend=2; n=tend/h; T=t; Y=y;
for i=1:n
    f=sin(t*y); y=y+h*f; t=t+h;
    T=[T; t]; Y=[Y; y];
end
fend=y; plot(T,Y,T,Y,'o');
return

>> h=0.2; Y2=[];
>> for k=0:3, Y2=[Y2 feuler(h/2^k)], end;

Y2 = 5.9379  5.9007  5.8823  5.8731
```



A Richardson extrapolation can be used to cancel the leading (global) error in $\mathcal{O}(h)$ and calculate the value $y(2) \approx 5.8731 + \frac{5.8731 - 5.8823}{2^Q - 1} = 5.8639$.

Peer Teaching (2×1 minutes to think, explain to your neighbour and vote)

Richardson extrapolation. Which value for Q should you use here above?

← 0

↑ 1

→ 2

Runge-Kutta (RK2) achieves a better precision with a global error in $\mathcal{O}(h^2)$

$$y_{i+1} = y_i + \frac{h}{2}(k_1 + k_2), \quad \text{with} \quad \begin{cases} k_1 = f(t_i, y_i) \\ k_2 = f(t_i + h, y_i + hk_1) \end{cases}$$

Runge-Kutta (RK4) achieves a high precision with a global error in $\mathcal{O}(h^4)$

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad \text{with} \quad \begin{cases} k_1 = f(t_i, y_i) \\ k_2 = f(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_1) \\ k_3 = f(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_2) \\ k_4 = f(t_i + h, y_i + hk_3) \end{cases}$$

In Matlab use `help ode23`, `ode45`, `odeset` for RK methods with variable step size, where the step size h is continually adjusted to achieve a specified precision with a minimum number of steps.

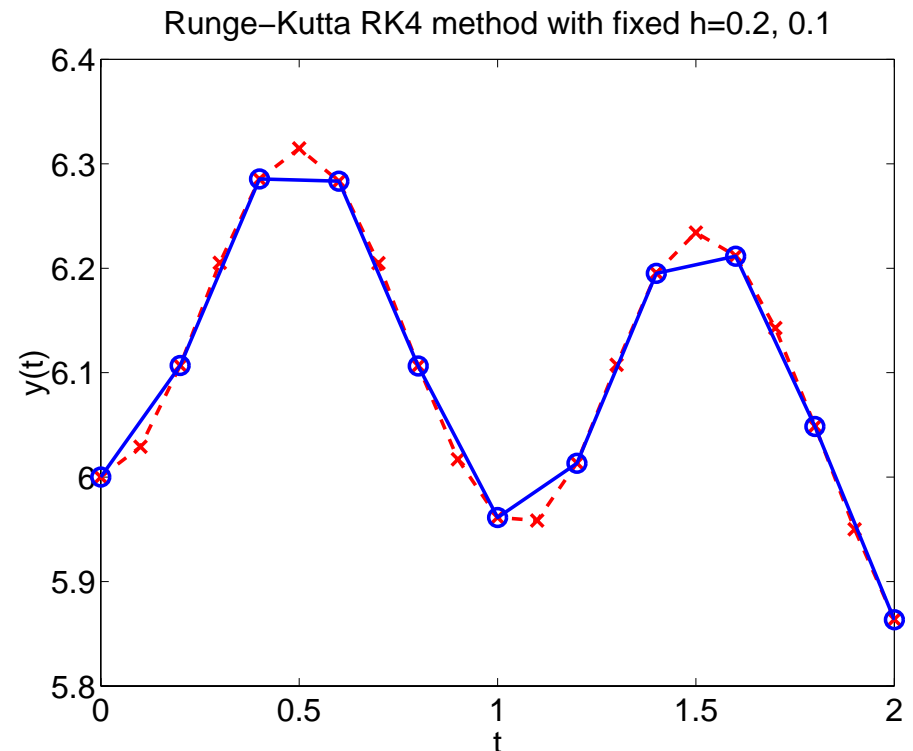
Example solve $y'(t) = \sin(ty)$ with $y(0) = 6$ using the RK4 method

```
function f=fsin(t,y),
    f=sin(t*y);
return

function fend=rk4sin(h)
    time=[0 34]; y=6.1866; t=time(1);
    n=diff(time)/h; T=t; Y=y; h2=h/2;
    for i=1:n
        k1=fsin(t, y);
        k2=fsin(t+h2,y+h2*k1);
        k3=fsin(t+h2,y+h2*k2);
        k4=fsin(t+h ,y+h *k3);
        y=y+h/6*(k1+2*k2+2*k3+k4);
        t=t+h;
        T=[T; t]; Y=[Y; y];
    end
    plot(T,Y); fend=y;
return
```

```
>> h=0.2; YN=[];
>> for k=0:1, YN=[YN rk4sin(h/2^k)]; Y2=YN(end), end;
Y2 = 5.86346010701075    5.86390480621220
```

```
>> % --- Alternative using Matlab's solvers
>> tend=2; y0=6; options=odeset('RelTol',1e-6,'AbsTol',1e-4]);
>> [T,Y]=ode23(@fsin,[0 tend],y0,options);
>> [T,Y]=ode45(@fsin,[0 tend],y0,options);
```



A Richardson extrapolation again cancels the leading (global) error, here in $\mathcal{O}(h^4)$ with $y(2) \approx 5.86390 + \frac{5.86390 - 5.86346}{2^4 - 1} = 5.86392$.

11.3 System of ODEs: predator-prey model (NAM 8.3, H ex9.4 cp9.1)

Example: Volterra-Lotka's *predator-prey* model is described by two coupled ODEs

$$\begin{cases} \frac{dy_1}{dt} = y_1(\alpha_1 - \beta_1 y_2) \\ \frac{dy_2}{dt} = y_2(-\alpha_2 + \beta_2 y_1) \end{cases} \Leftrightarrow y' = \begin{pmatrix} \alpha_1 y_1 - \beta_1 y_1 y_2 \\ -\alpha_2 y_2 + \beta_2 y_1 y_2 \end{pmatrix} = f(y)$$

where y_1, y_2 are the number of preys (e.g. fish) and predators (e.g. sharks). Take $\alpha_1 = 1.0$ is the natural growth rate of the prey population

$\beta_1 = 0.1$ is the rate at which preys are eaten by the predators

$\alpha_2 = 0.5$ is the natural death rate of the predators

$\beta_2 = 0.02$ is the rate at which the predators grow given an amount of food y_1

This is an *autonomous* system, i.e. time does not appear in the right hand side $f = f(y)$; it can be solved using a RK4 or Matlab's own solver `ode45`.

Using Matlab `ode45`, the time has to be defined as an argument even if it is not used

```
function f=fpp(t,y) % arguments
    a1=1.; b1=0.1; a2=0.5; b2=0.02; % parameters
    f=[ a1*y(1)-b1*y(1)*y(2)
        -a2*y(2)+b2*y(1)*y(2)];

>> time=[0 22]; y0=[100 10]; [T,Y]=ode45('fpp',time,y0);
>> figure(1);plot(T,Y);
>> figure(2);plot(Y(:,1),Y(:,2),y0(1),y0(2),'o')
```

Using your own RK4

```
>> time=[0 22]; y0=[100 10]'; h=0.1;
>> t=time(1); y=y0; T=t; Y=y
>> for i=1:diff(time)/h
>>   f1=fpp(t,y);      f2=fpp(t,y+h*f1/2);
>>   f3=fpp(t,y+h*f2/2); f4=fpp(t,y+h*f3);
>>   y=y+h/6*(f1+2*f2+2*f3+f4); t=t+h;
>>   T=[T t]; Y=[Y y];
>> end
>> plot(T,Y)
```

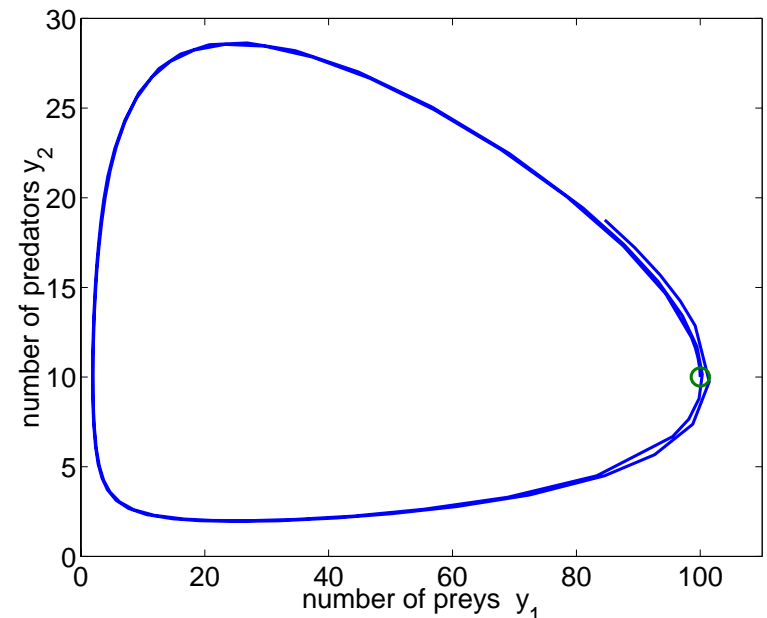
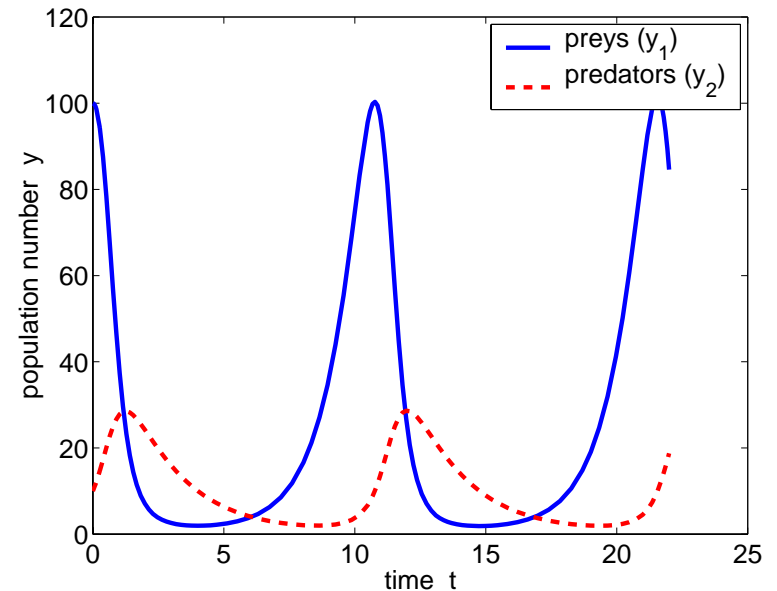
The solution has a period of $T \approx 10$ depending on the initial condition.

Also note the existence of a stationary point $(\alpha_2/\beta_2; \alpha_1/\beta_1) = (25, 10)$.

Peer Teaching:

A fishy problem. Account for the presence of fishermen; does the absolute maximum number of preys

↑ increase ↔ stay same ↓ decrease



11.4 System of ODEs: high order equations (NAM 8.4, H 9.1)

High order ODEs can always be transformed into a system of 1st order equations

Define new unknowns $u_1(t) = y(t)$, $u_2(t) = y'(t)$, $u_3(t) = y''(t)$, etc, and solve the system

$$\mathbf{u}' = \begin{pmatrix} u_1' \\ u_2' \\ \dots \\ u_{k-1}' \\ u_k' \end{pmatrix} = \begin{pmatrix} u_2 \\ u_3 \\ \dots \\ u_k' \\ f(t, u_1, u_2, \dots, u_k) \end{pmatrix} = \mathbf{g}(t, \mathbf{u})$$

Example pendulum with large oscillations.

$$\frac{d^2\varphi}{dt^2} + c \frac{d\varphi}{dt} \left| \frac{d\varphi}{dt} \right| + \frac{g}{L} \sin \varphi = 0, \quad \text{with} \quad \varphi(0) = \pi, \quad \varphi'(0) = \dot{\varphi}_0$$

Define $u_1 = \varphi$, $u_2 = \varphi'$ to obtain an equivalent first order system of ODEs

$$\mathbf{u}' = \begin{pmatrix} u_1' \\ u_2' \end{pmatrix} = \begin{pmatrix} u_2 \\ -\frac{g}{L} \sin u_1 - cu_2|u_2| \end{pmatrix}$$

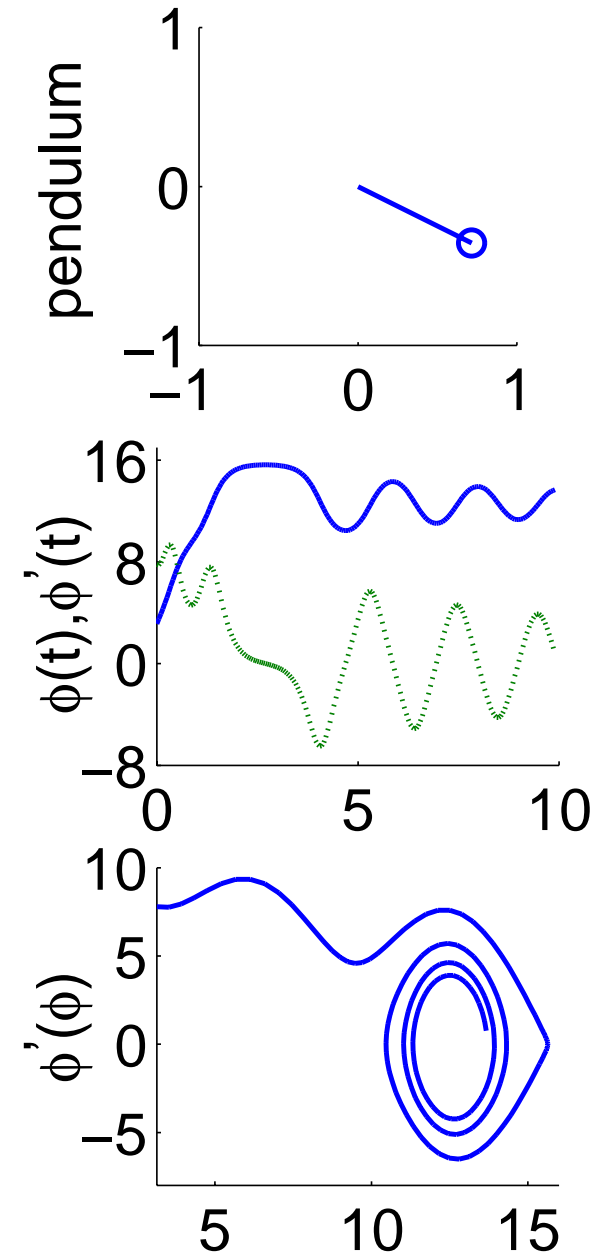
The solution obtained with RK4 can be plotted dynamically with

```

function f=fpend(t,u)                                % arguments
global L c g                                        % parameters
f=[ u(2)
    -g/L*sin(u(1))-c*u(2)*abs(u(2))];

global L c g; clf                                  % init
L=0.8; c=0.05; g=9.81; dt=0.05;                   % parameters
y0=[pi 7.8]; time=[0 9.9];                         % IC, step
t=0; y=y0; Y=y; T=t;
%
xL=L*sin(y(1)); yL=-L*cos(y(1));                    % plot
subplot(3,3,7);hold on;ylabel('\phi^{\prime}(\phi)');
subplot(3,3,4);hold on;ylabel('\phi(t), \phi^{\prime}(t)')
subplot(3,3,1),hold on;ylabel('pendulum');
    axis(1.25*[-L L -L L]),axis square
    plot([0 xL],[0 yL],'-'),plot(xL,yL,'o'),
    set(gcf,'DoubleBuffer','on')                    % dyn plot
%
while t<time(2)-dt/2
    f1=fpend(y);          f2=fpend(y+dt*f1/2);% RK4
    f3=fpend(y+dt*f2/2);f4=fpend(y+dt*f3);
    y=y+dt*(f1+2*f2+2*f3+f4)/6;  t=t+dt;
    Y=[Y; y]; T=[T; t]; pause(0.0)                % dyn plot
    subplot(3,3,1);plot([0 xL],[0 yL], 'w-'),plot(xL,yL,'wo')
    xL=L*sin(y(1)); yL=-L*cos(y(1));
    subplot(3,3,1),plot([0 xL],[0 yL],'-'), plot(xL,yL,'o')
    subplot(3,3,4),plot(T(end-1:end),Y(end-1:end,1),...
        T(end-1:end),Y(end-1:end,2),':')
    subplot(3,3,7), plot(Y(end-1:end,1),Y(end-1:end,2))
end

```

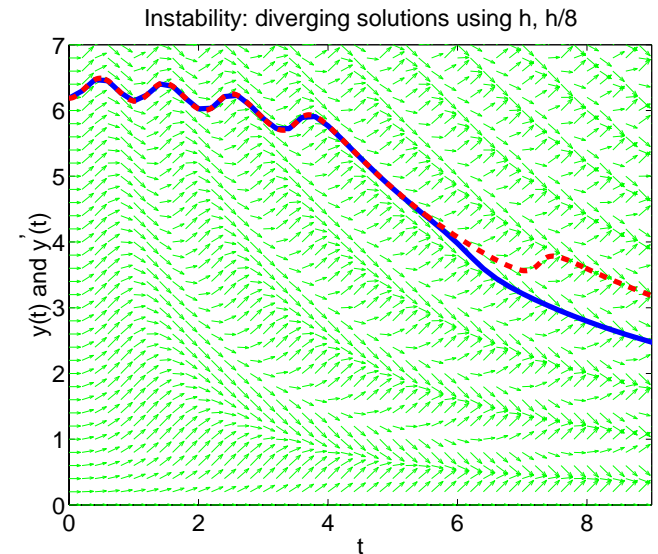
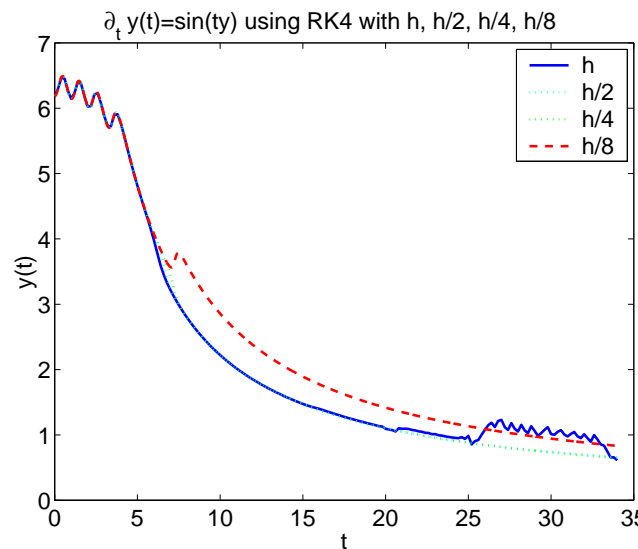


11.5 Warning: instabilities, stiff problems (NAM 8.6, H 9.3)

Instabilities appear when discretization errors get amplified

Reducing gradually the step size with $h, h/2, h/4$, RK4 appears to converge to the green dotted line; with $h/8$ the red dashed solution is totally different!

```
>> t=0:0.2:34; y=0:0.2:7;
>> [T,Y]=meshgrid(t,y);
>> FT=0.01*ones(size(T));
>> FY=0.01*sin(T.*Y);
>> quiver(T,Y,FT,FY);
>> axis([0 9 0 7])
>> for k=0:2
>>   rk4sin(0.2/2^k)
>> end
ans = 0.6123
      0.6556
      0.6474
      0.8323
```



Two neighbouring solutions grow exponentially apart; the cure is to use *implicit methods* taught in follow-up courses (TILLNUM 1&2)

Stiff problems involve two different scales; the step size is dictated by the small scale

$$y' = -1000y + 1000t + 1001 \quad \text{with} \quad y(0) = 1$$

$$\text{general solution:} \quad y(t) = 1 + t + ce^{-1000t}$$

$$\text{particular solution:} \quad y(t) = 1 + t$$

Euler & RK methods require $h < 0.002$ to reproduce this simple linear solution.