UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KATEDRA ZA AUTOMATIKU I UPRAVLJANJE SISTEMIMA

Obične diferencijalne jednačine
Numeričko rešavanje

Modeliranje i simulacija sistema

Obične diferencijalne jednačine (ODJ)

- engl. *Ordinary Differential Equation* (ODE)
- U matematici ODJ je jednačina koju čini funkcija jedne nezavisne promenljive i njenih izvoda.
- Veoma su česte u modelovanju dinamičkih sistema
  - U simulacijama nezavisno promenljiva je vreme *t*

Linearna ODJ

- Linearna ODJ sadrži linearnu kombinaciju izvoda zavisno promenljive *y*

$$y^{(n)} = \sum_{i=0}^{n-1} a_i(t) \cdot y^{(i)} + r(t)$$

ili

$$y^{(n)} = a_{n-1} y^{(n-1)} + \ldots + a_1 y' + a_0 y + r(t)$$

- Linearne ODJ se rešavaju na poseban način koji je dosta jednostavniji od opšteg postupka.

Svođenje ODJ na sistem DJ 1. reda

- Svaka ODJ se može napisati kao ekvivalentan sistem običnih diferencijalnih jednačina 1. reda.
- Jedan od načina za takvu transformaciju je uvođenje smena

$$y_k = y^{(k-1)}, \quad k = 1,2,\ldots,n$$

- Dobija se:

$$y_1' = y_2$$
$$y_2' = y_3$$
$$\ldots$$
$$y_{n-1}' = y_n$$
$$y_n' = f(t, y_1, \ldots, y_n)$$

ili zapisano u vektorskom obliku:

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ y_n \end{bmatrix}, \mathbf{y}' = \begin{bmatrix} y_1' \\ y_2' \\ \ldots \\ y_n' \end{bmatrix}, \mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ \ldots \\ f_n \end{bmatrix} = \begin{bmatrix} y_2 \\ y_3 \\ \ldots \\ f(t, y_1, \ldots, y_n) \end{bmatrix}$$

Primer transformacije

- Posmatra se ODJ:

$$\frac{d^2y}{dt^2} - \mu(1-y^2)\frac{dy}{dt} + y = 0$$

Smene:

$$\left.\begin{aligned} y_1 &= y \\ y_2 &= \frac{dy}{dt} \end{aligned}\right\} \xrightarrow{\frac{d}{dt}} \begin{aligned} \frac{dy_1}{dt} &= \frac{dy}{dt} \\ \frac{dy_2}{dt} &= \frac{d^2y}{dt^2} \end{aligned} \longrightarrow \begin{aligned} \frac{dy_1}{dt} &= y_2 \\ \frac{dy_2}{dt} &= \mu(1-y_1^{\,2})y_2 - y_1 \end{aligned}$$

Uopšten sistem DJ 1. reda

- Sistem *n* običnih diferencijalnih jednačina prvog reda, gde se pojavljuje *n* zavisno promenljivih y_1,…, y_n

$$\frac{dy_1}{dt} = f_1(y_1, y_2, \ldots, y_n, t), \quad y_1(t_0) = y_{10}$$

$$\frac{dy_2}{dt} = f_2(y_1, y_2, \ldots, y_n, t), \quad y_2(t_0) = y_{20}$$

$$\ldots$$

$$\frac{dy_n}{dt} = f_n(y_1, y_2, \ldots, y_n, t), \quad y_n(t_0) = y_{n0}$$

Numeričko rešavanje ODJ

- Svodi se na rešavanje sistema diferencijalnih jednačina 1. reda
- Šire posmatrano, sistem običnih diferencijalnih jednačina se takođe svodi na sistem diferencijalnih jednačina 1.

- Primer:

$$m_1 x_1'' + c_1 x_1' + k_1 x_1 + k(x_1 - x_2) = f(t)$$
$$m_2 x_2'' + c_2 x_2' + k_2 x_2 + k(x_2 - x_1) = 0$$

$$y_1 = x_1 \qquad y_1' = x_1' = y_2$$
$$y_2 = x_1' \qquad y_2' = x_1'' = \frac{1}{m_1}\left(f(t) - c_1 y_2 - k_1 y_1 - k(y_1 - y_3)\right)$$
$$y_3 = x_2 \qquad y_3' = x_2' = y_4$$
$$y_4 = x_2' \qquad y_4' = x_2'' = \frac{1}{m_2}\left(-c_2 y_4 - k_2 y_3 - k(y_3 - y_1)\right)$$

Jednostavniji problem ...

- Problem:

$$\frac{dy}{dt} = f(t), \quad y(t_0) = y_0$$

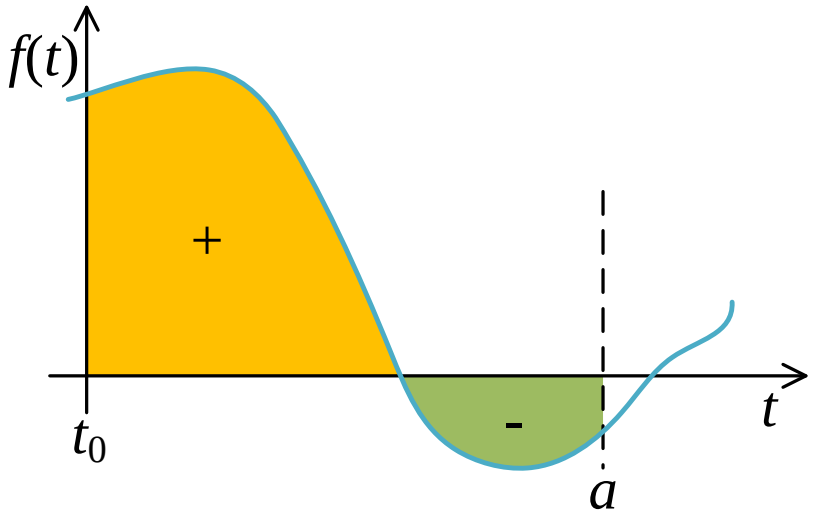
ima analitičko rešenje:

$$y(t) = y_0 + \int_{t_0}^{t} f(t)dt$$

- Konkretna vrednost

$$y(a) = \int_{0}^{a} f(t)dt, \quad t_0 = 0$$

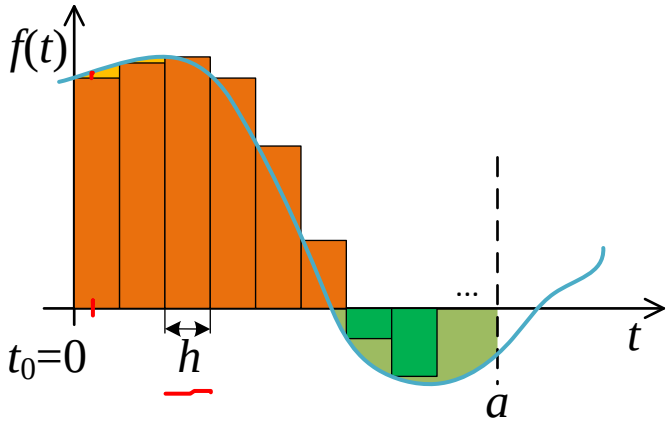
predstavlja površinu sa slike



Ideja numeričkog računanja

- Odredimo vrednosti funkcije $f(t)$ u diskretnim trenucima ih, i=0,1,2,…
- Vrednost integrala se može aproksimirati sumom pravougaonika sa slike

$$y(a) \approx \sum_{i=0}^{N-1} f(ih) \cdot h$$



- Ovakva računica je veoma gruba i **praktično se ne koristi**.
- Ako se h smanji greška se smanjuje, ali se produžava vreme računanja.

Numeričko rešavanje DJ 1. reda

Problem:

$$\frac{dy}{dt} = f(y,t), \quad y(t_0) = y_0$$

Rešenje:

- polazi se od (t_0, y_0) i sukcesivno računaju parovi (t_i, y_i) za i=1,2,…

- Iterativan račun se sprovodi “korak-po-korak”, gde se (t_{i+1}, y_{i+1}) računa na osnovu prethodno sračunatog (t_i, y_i).

Razmatrani algoritmi

- Ojlerov postupak
- Metode Runge-Kuta

Leonhard Euler

Born	15 April 1707 Basel, Switzerland
Died	18 September 1783 (aged 76) [OS: 7 September 1783] Saint Petersburg, Russian Empire
Residence	Kingdom of Prussia, Russian Empire Switzerland
Nationality	Swiss
Fields	Mathematics and physics
Institutions	Imperial Russian Academy of Sciences Berlin Academy
Alma mater	University of Basel
Doctoral advisor	Johann Bernoulli
Doctoral students	Nicolas Fuss Johann Hennert Joseph Louis Lagrange Stepan Rumovsky

Ojlerov postupak

- Vrednost y_{i+1} se može odrediti na osnovu razvoja u Tejlorov red u okolini tačke (t_i, y_i)

$$y_{i+1} = y_i + \Delta y = y_i + \frac{dy_i}{dt} h + \frac{d^2 y_i}{dt^2} \frac{h^2}{2!} + \ldots \qquad h = t_{i+1} - t_i$$

- Za h malo y_{i+1} se aproksimira sa

$$y_{i+1} \approx y_i + \frac{dy_i}{dt} h$$

- Nakon diferenciranja se dobija

$$\frac{dy_{i+1}}{dt} = \frac{dy_i}{dt} + \frac{d^2 y_i}{dt^2} h \qquad \text{ili} \qquad \frac{d^2 y_i}{dt^2} = \left(\frac{dy_{i+1}}{dt} - \frac{dy_i}{dt} \right) \frac{1}{h}$$

Ojlerov postupak (2)

$$\frac{d^2 y_i}{dt^2} = \left(\frac{dy_{i+1}}{dt} - \frac{dy_i}{dt} \right) \frac{1}{h}$$

- Smenom u gornji izraz se dobija bolja aproksimacija

$$y_{i+1} = y_i + \frac{dy_i}{dt} h + \frac{d^2 y_i}{dt^2} \frac{h^2}{2!} = y_i + \frac{dy_i}{dt} h + \left(\frac{dy_{i+1}}{dt} - \frac{dy_i}{dt} \right) \frac{h}{2}$$

Računanje izvoda traži vrednost y_{i+1} koju još ne znamo!

Ojlerov postupak (3)

U svakoj iteraciji :

1. Uvodi se predviđena (*predicted*) vrednost

$$y_{i+1}^{p} = y_i + \frac{dy_i}{dt} h$$

← metod 1. reda
(ako se ne radi drugi korak)

2. I ona se koriguje

$$\varepsilon_i = \left(\frac{dy_{i+1}^{p}}{dt} - \frac{dy_i}{dt} \right) \frac{h}{2}$$

← Korekcija, a ujedno
i procena greške

$$y_{i+1}^{c} = y_{i+1}^{p} + \varepsilon_i$$

← metod 2. reda

Ojlerov postupak - realizacija

U tekućoj iteraciji …

- indeks *i*+1 je zamenjen sa 1
- prethodna iteracija ima indeks 0

$$y_1^p = y_0 + \frac{dy_0}{dt} h$$

$$\varepsilon = \left(\frac{dy_1^p}{dt} - \frac{dy_0}{dt} \right) \frac{h}{2}$$

$$y_1^c = y_1^p + \varepsilon$$

```
function OjlerKorak(f, t0, y0, h)
    dy0 = f(t0,y0);       # izvod u tacki (t0,y0)
    y1p = y0 + dy0*h;
    t1 = t0 + h;
    dy1 = f(t1,y1p);    # izvod u tački (t1,y1p)
    e = (dy1-dy0)*h/2.0;          # procena greške
    y1c = y1p + e;        # korekcija
    return t1, y1c, e
end;
```

Rešenje nad intervalom [0, t_k] sa korakom *h* …

```
function Ojler2(f, tk, y0, h)
    t0 = 0;
    Y = [y0];
    koraka = tk / h;
    for i = 1:koraka
        t1, y1, e = OjlerKorak(f,t0,y0,h);
        y0 = y1;
        t0 = t1;
        push!(Y, y1);     # zapamti y1
    end
    return Y
end;
```

Primer: DJ 1. reda

Problem:

$$\frac{dy}{dt} = \lambda \cdot e^{-\alpha \cdot t} \cdot y$$

$$y(0) = y_0$$

```
function difJedn(t,y)
    α = 1; λ = 1;
    dy = λ*exp(-α*t)*y;
    return dy
end;
```

```
function analitickoResenje(t,y0)
    α = 1; λ = 1;
    y = y0*exp(λ/α*(1-exp(-α*t)));
    return y
end;
```

Analitičko rešenje:

$$y(t) = y_0 \cdot e^{\frac{\lambda}{\alpha} \cdot \left(1 - e^{-\alpha \cdot t}\right)}$$

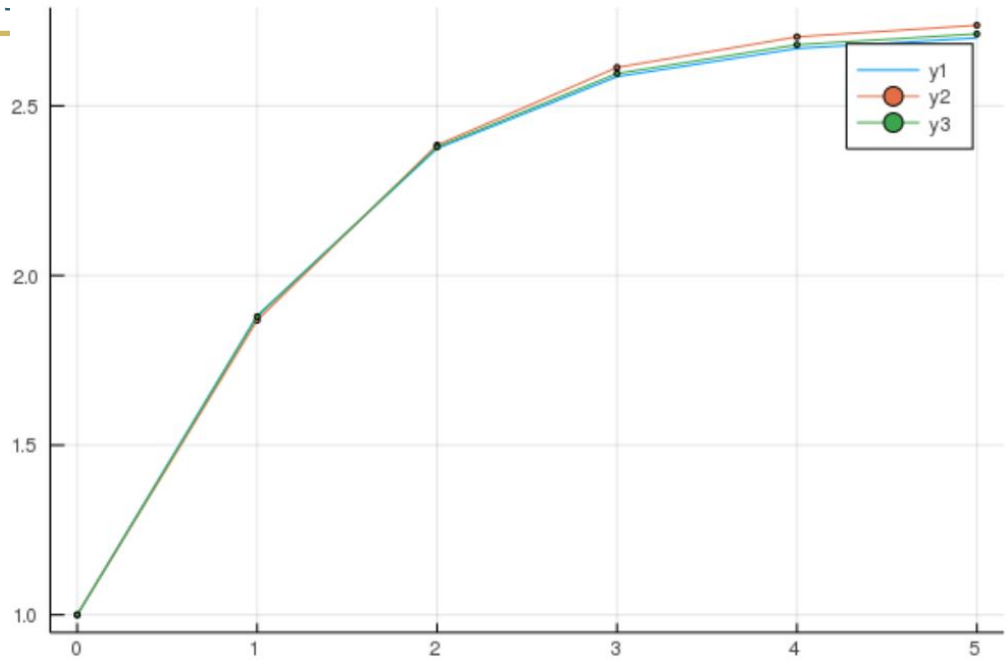
```
tkraj = 5;
t = (0:tkraj);
y0 = 1.0;
ya = analitickoResenje.(t,y0);

y1  = Ojler2( difJedn, tkraj, y0, 1 );          # h=1
y05 = Ojler2( difJedn, tkraj, y0, 0.5 );    # h=0.5

y05 = y05[1:2:end];       # izbaci rešenja u 0.5s, 1.5s, …
[t ya y1 y1-ya y05 y05-ya]
```

Primer: rezultati

t	ya	ye1	ye1-ya	ye05	ye05-ya
0.0	1.0000	1.0000	0	1.0000	0
1.0	1.8816	1.8679	-0.0137	1.8786	-0.0030
2.0	2.3742	2.3843	0.0101	2.3786	0.0044
3.0	2.5863	2.6131	0.0268	2.5952	0.0090
4.0	2.6689	2.7033	0.0343	2.6799	0.0110
5.0	2.7000	2.7373	0.0373	2.7118	0.0117

Legenda:
ya – analitičko rešenje
ye1 – Euler 2. reda sa fiksnim korakom h=1
ye05 – Euler 2. reda sa fiksnim korakom h=0.5

```
using Plots
g1 = plot(t, ya)
g2 = plot!(g1, t, y1, marker=2)
g3 = plot!(g2, t, y05, marker=2)
```



Promenljivi korak integracije

- Računanje korekcije ε zavisi od koraka *h*
- Ideja: na osnovu veličine ε prilagoditi korak *h*
  - za veliko ε smanjiti korak, a
  - za malo ε povećati korak

$$\varepsilon_i = \left(\frac{dy_{i+1}^{p}}{dt} - \frac{dy_i}{dt} \right) \frac{h}{2}$$

- Parametri proračuna:
  - Dozvoljena relativna greška *relGr*
  - Dozvoljena apsolutna greška *apsGr*
  - Minimalan korak h_{min}

Promenljivi korak integracije (2)

```
function Ojler2ad(f,tk,y0,hmin,apsGr,relGr)
    t0 = 0.0;
    Y = [y0]; T = [t0];
    h = (tk-t0)/8.0;  # pocetni korak

    while t0 < tk
        if t0+h > tk
            h = tk - t0;    # ogranici poslednji korak
        end
        t1, y1, e = OjlerKorak(f,t0,y0,h);
        korakDobar = true;   # fleg za kraj koraka integracije

        # Ako je greska velika smanji korak i resetuj fleg
        if h > hmin
            if abs(e) > abs(y1)*relGr+apsGr
                h = h / 2.0;
                korakDobar = false;
            end
            if h < hmin
                h = hmin;
            end    # ogranici min korak
        end
    …
```

```
        …
        if korakDobar   # Nema greske: nastavi ka novoj tacki
            y0 = y1; t0 = t1;
            push!(Y, y1);
            push!(T, t1);
            # da li se korak moze povecati?
            if abs(e) < (abs(y1)*relGr+apsGr)/4.0
                h = h * 2.0;
            end
        end
    end
    return T, Y
end;
```

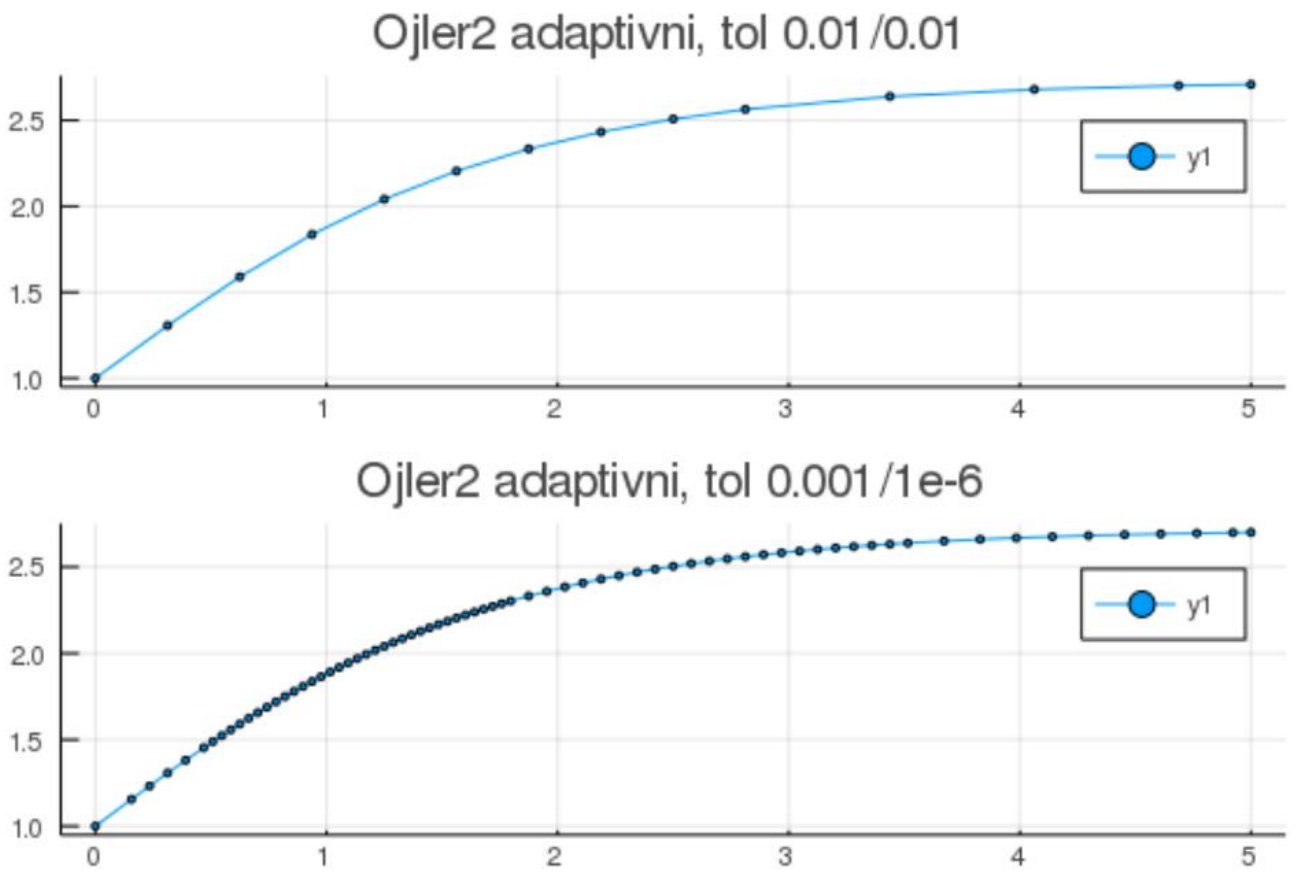
Primer: rezultati sa promenljivim korakom

```
tkraj = 5;
t = (0:tkraj);
y0 = 1.0;
ya = analitickoResenje.(t,y0);

t3, y3 = Ojler2ad(difJedn,tkraj,y0,0.01,0.01,0.01);
t4, y4 = Ojler2ad(difJedn,tkraj,y0,0.01,0.001,0.000001);

[tkraj ya[end] y3[end] y3[end]-ya[end] y4[end] y4[end]-ya[end]]
```

```
    t      ya        y3i       y3i-ya    y4i       y4i-ya
-------------------------------------------------------------
   5.0  2.70003    2.7085    0.00846809 2.6998   -0.000225091
Legenda:
ya  – analitičko rešenje
y3i – Euler 2. reda sa promenljivim korakom: h=0.01 relGr=0.01  apsGr=0.01
y4i – Euler 2. reda sa promenljivim korakom: h=0.01 relGr=0.001 apsGr=0.000001
```

Primer …

```
g1 = plot(t3, y3, marker=2, title="Ojler2 adaptivni, tol 0.01/0.01")
g2 = plot(t4, y4, marker=2, title="Ojler2 adaptivni, tol 0.001/1e-6")
plot(g1, g2, layout=(2,1))
```



Metode Runge-Kuta (RK)

- Runge i Kuta su uopštili Ojlerov postupak
- Pored toga, uveli su aproksimacije za više članove razvoja *y* u Tejlorov red
  - Ako se aproksimiraju članovi do 2. izvoda onda se govori o metodama 2. reda (postoji čitava familija tih metoda)
  - Ako se aproksimiraju članovi do 3. izvoda onda se govori o metodama 3. reda (opet, više metoda)
  - …

Carl Runge

Born	30 August 1856 Bremen, German Confederation
Died	3 January 1927 (aged 70) Göttingen, Weimar Republic
Residence	Germany
Citizenship	German
Fields	Mathematics Physics
Institutions	University of Hannover (1886–1904) Georg-August University of Göttingen (1904–1925)
Alma mater	Berlin University
Doctoral advisor	Karl Weierstrass Ernst Kummer
Doctoral students	Max Born
Known for	Runge–Kutta method Runge's phenomenon Laplace–Runge–Lenz vector

Martin Wilhelm Kutta

Born	November 3, 1867 Pitschen, Upper Silesia
Died	December 25, 1944 (aged 77) Fürstenfeldbruck
Residence	Germany
Nationality	German
Fields	Mathematician
Institutions	University of Stuttgart RWTH Aachen
Alma mater	University of Breslau University of Munich
Doctoral advisor	C. L. Ferdinand Lindemann Gustav A. Bauer
Other academic advisors	Walther Franz Anton von Dyck
Known for	Runge-Kutta method Zhukovsky-Kutta aerofoil Kutta–Joukowski theorem Kutta condition

RK 2. reda

- Ojlerov metod – ponovo …

$$y_{i+1}^{p} = y_i + f(y_i, t_i)h = y_i + k_1$$

$$\varepsilon_i = \left(f(y_i + k_1, t_i + h) - f(y_i, t_i)\right)\frac{h}{2} = \frac{k_2 - k_1}{2}$$

$$y_{i+1}^{c} = y_i + k_1 + \frac{k_2 - k_1}{2} = y_i + \frac{1}{2}k_1 + \frac{1}{2}k_2$$

- Može se zapisati kao

$$y_{i+1} = y_i + \frac{1}{2}k_1 + \frac{1}{2}k_2$$

$$k_1 = f(y_i, t_i)h$$

$$k_2 = f(y_i + k_1, t_i + h)h$$

RK uvodi tri konstante c_1, c_2, a_2

$$y_{i+1} = y_i + c_1 k_1 + c_2 k_2$$

$$k_1 = f(y_i, t_i)h$$

$$k_2 = f(y_i + a_2 k_1, t_i + a_2 h)h$$

RK 2. reda

- Može se pokazati da između c_1, c_2, a_2 vrednosti postoje zavisnosti (izvođenje postoji u odvojenom materijalu)

$$c_1 + c_2 = 1 \qquad c_2 a_2 = \frac{1}{2}$$

- Ovaj sistem 3 jednačine sa 2 nepoznate daje slobodu izbora rešenja.

- Jedno moguće rešenje vodi ka Ojlerovom metodu 2. reda

$$c_1 = \frac{1}{2}, \quad c_2 = \frac{1}{2}, \quad a_2 = 1$$

- Postoje i druga rešenja, npr.

$$c_1 = 0, \quad c_2 = 1, \quad a_2 = \frac{1}{2} \qquad c_1 = \frac{1}{4}, \quad c_2 = \frac{3}{4}, \quad a_2 = \frac{2}{3}$$

U tim rešenjima se vrednosti *f* računaju unutar intervala *h.*

RK 3. reda

- Uvode se konstante: c_1, c_2, c_3, a_2, a_3, b_3

$$y_{i+1} = y_i + c_1 k_1 + c_2 k_2 + c_3 k_3$$

$$k_1 = f(y_i, t_i)h$$

$$k_2 = f(y_i + a_2 k_1, t_i + a_2 h)h$$

} Isto kao kod RK 2. reda

$$k_3 = f(y_i + b_3 k_1 + (a_3 - b_3)k_2, t_i + a_3 h)h$$

- Veze konstanti

$$c_1 + c_2 + c_3 = 1$$
$$c_2 a_2 + c_3 a_3 = 1/2$$
$$c_2 a_2^2 + c_3 a_3^2 = 1/3$$
$$c_3 (a_3 - b_3) a_2 = 1/6$$

Moguće rešenje

$$c_1 = 2/8$$
$$c_2 = c_3 = 3/8$$
$$a_2 = a_3 = 2/3$$
$$b_3 = 0$$

RK 3. reda - realizacija

```
function rkutta3( F, tp, tf, y0, h )
    # Metod integracije RUNGE-KUTTA 3. reda sa nepromenljivim korakom

    T = tp : h : tf;
    n = length(T);
    Y = zeros( n, length(y0) );
    Y[1,:] = y0[:]';
    t = tp;
    for i = 1 : n-1
        y = Y[i,:]';
        k1 = h * F( t, y )
        k2 = h * F( t+2/3*h, y+2/3*k1 );
        k3 = h * F( t+2/3*h, y+2/3*k2 );
        Y[i+1,:] = (y + k1/4 + 3/8*k2 + 3/8*k3)';
        t += h;
    end
    return T, Y
end;
```

Primer: rezultati RK 3

```
t = (0:1.0:tkraj);
y0 = 1.0;
ya = analitickoResenje.(t,y0);
y1 = Ojler2(difJedn,tkraj,y0,1.0);

t3, y3 = rkutta3(difJedn,0.0,tkraj,[y0],1.0);          # h=1
[collect(t) ya y1 y1-ya y3 y3-ya]
```

t	ya	y1	y1-ya	y3	y3-ya
0.0	1.0000	1.0000	0	1.0000	0
1.0	1.8816	1.8679	-0.0137	1.8732	-0.0083
2.0	2.3742	2.3843	0.0101	2.3642	-0.0100
3.0	2.5863	2.6131	0.0268	2.5761	-0.0102
4.0	2.6689	2.7033	0.0343	2.6588	-0.0101
5.0	2.7000	2.7373	0.0373	2.6899	-0.0101

Legenda:
ya – analitičko rešenje
y1 – Euler 2. reda sa fiksnim korakom: h=1
y3 – RK 3. reda sa fiksnim korakom: h=1

RK 4-5. reda

```
using LinearAlgebra

function rkutta45( F, tp, tf, y0, hmin, absGr, relGr )
    # Metod integracije RUNGE-KUTTA 4-5 reda, ima promenljivi korak
    t = tp;     T = [t];
    y = y0[:];  Y = [y'];
    h = (tf-tp)/8.0;        # početni korak

    while t < tf
        if t+h > tf
            h = tf-t;
        end                 # ograniči poslednji korak

        k1 = h * F(
        k2 = h * F(
        k3 = h * F(
        k4 = h * F(
        k5 = h * F(
        k6 = h * F(
        y4 = y + 25/
        y5 = y + 16/
        e = y5 - y4
        korakDobar =
```

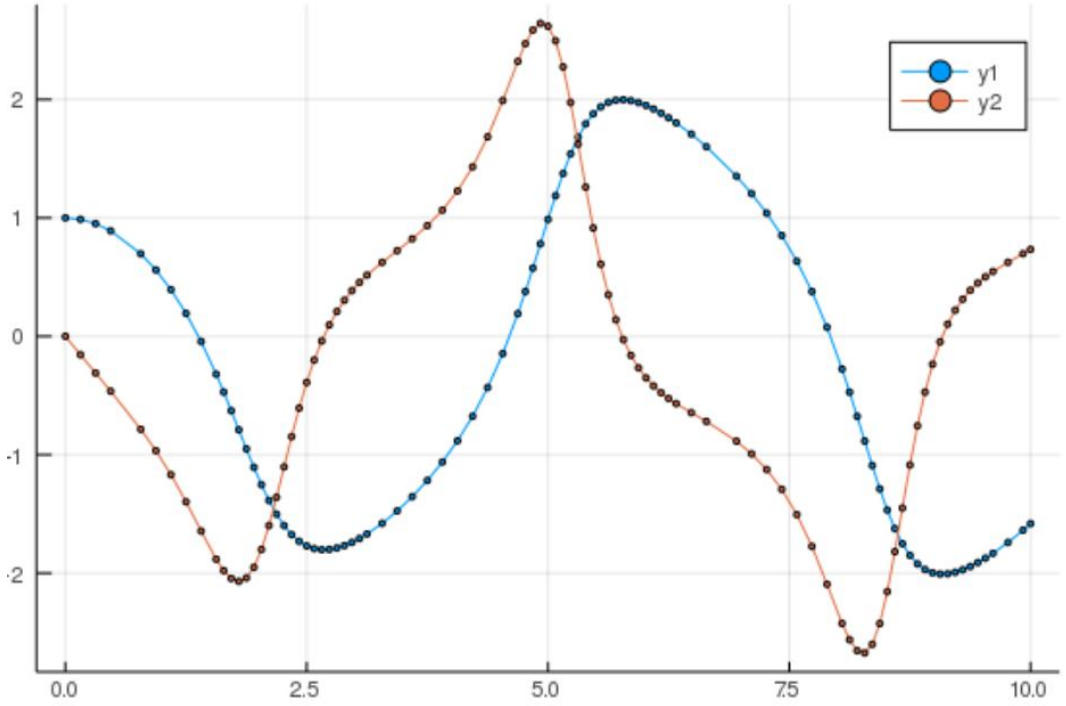
```
    if h > hmin         # Ako je greška velika smanji korak i resetuj fleg
        if all(norm(e) > max(norm(y5)*relGr,absGr))
            h = h / 2.0;
            korakDobar = false;
        end
        if h < hmin
            h = hmin;
        end    # ogranici min korak
    end
    if korakDobar       # Nema greske ? nastavi ka novoj tacki (po vremenu)
        y = y5; t += h;
        push!(Y, y');
        push!(T, t);
        if all(norm(e) < max(norm(y5)*relGr,absGr)/4.0)
            h = h * 2.0;
        end                 # povecati korak?
    end
  end
  return T, Y
end;
```

Primer RK 4-5. reda

```
function vdpol( t, x )
    # Opis Van der Pol-ove dif. jednacine
    x1, x2 = x
    xp = [x2;
          (1-x1^2)*x2-x1 ];
    return xp
end;
```

$$\frac{d^2x}{dt^2} - \mu(1-x^2)\frac{dx}{dt} + x = 0$$

```
t, y = rkutta45(vdpol, 0.0, 10.0, [1.0;0.0], 1e-6, 1e-6, 1e-6);

z = [a[i] for a in y, i in 1:2]
plot(t,z, marker=2)
```



MATLAB i rešavanje ODJ

- MATLAB sadrži ODE metode za rešavanje sistema ODJ 1. reda: ode23, ode45, ode113, ode15s, ode23s, ode23t, ode23tb, ode15i

- Sintaksa: **[T, Y] = solver(difJedn, vremenskiOpseg, y0, opcije)**

  solver - je neki od algoritama ode23, ode45, …

  difJedn - je naš opis jednačina koje se rešavaju

  vremenskiOpseg

  - Tk – krajnje vreme (početno vreme ==0)
  - [Tp Tk] – interval u kome se traži rešenje
  - [Tp t1 t2 t3 … Tk] – zadati trenuci u kojima se očekuje rešenje

  y0 – vrednosti zavisno promenljivih u početnom trenutku

  opcije – podešavanje parametara ODE algoritma

  - Postavlja se funkcijom **odeset**
  - Podešava: apsolutnu grešku, relativnu grešku, početni korak, minimalan korak, …

  T – vektor vremenskih trenutaka rešenja

  Y – matrica vrednosti zavisno promenljivih u trenucima T

Osobine solvera

Naziv	Metod	*Stiff*	Tačnost	Kada upotrebiti?
ode45	Runge-Kutta (4,5)	ne	srednja	uglavnom uvek. Prvo njega probati!
ode23	Runge-Kutta (2,3)	ne	mala	za manje tačno računanje i kada je model blago *stiff*
ode113	Adams-Bashforth-Moulton	ne	mala do velika	za tačno računanje numerički zahtevnih problema
ode15s	NDF	da	mala do srednja	kada je ode45 spor zbog *stiff*
ode23s	modifikovan Rosenbrock	da	mala	za manje tačno računanje *stiff*
ode23t	trapezoidno pravilo	srednje da	mala	Za srednje *stiff* probleme
ode23tb	TR-BDF2	da	mala	za manje tačno računanje *stiff*

Kod *stiff* (krutih) problema neke promenljive stanja se mogu menjati vrlo sporo u odnosu na interval simulacije, a druge mnogo brže. Metode koje nisu dizajnirane za *stiff* probleme nisu efikasne jer primenjuju mali korak integracije.

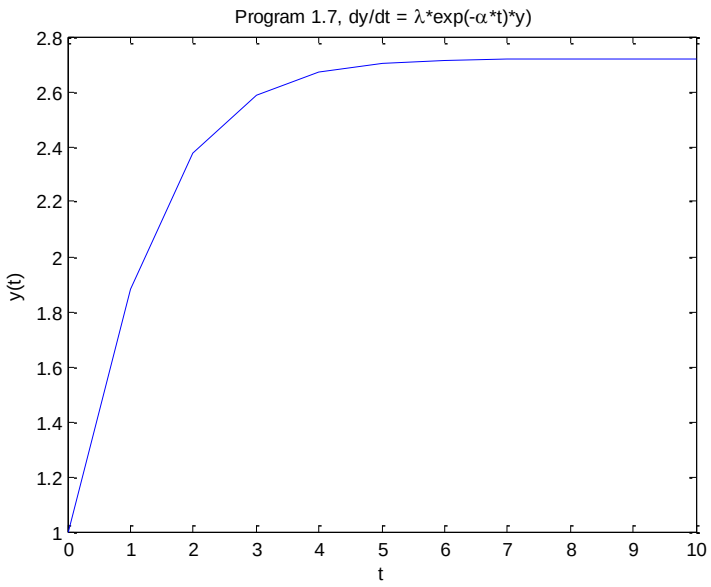
Osobine solvera (2)

- **ode45** zasnovan na Runge-Kutta (4,5) metodi (Dormand-Prince)
  - Jednokoračan - računa $y(t_n)$ u jednom koraku na osnovu $y(t_{n-1})$
  - Pogodan za brojne probleme sa kontinulanim stanjima
  - Podrazumevan algoritam
- **ode23** zasnovan na Runge-Kutta (2,3) (Bogacki & Shampine)
  - Sličan ode45, ali može biti efikasniji za manje tačno računanje i kada je model blago *stiff*
- **ode113** je metod promenljivog reda (Adams-Bashforth-Moulton)
  - Može biti efikasniji od ode45 za tačnije računanje
  - Višekoračan - računa $y(t_n)$ na osnovu više prethodnih vrednosti
- **ode15s** je metod promenljivog reda zasnovan na formulama numeričkog diferenciranja (NDF).
  - Višekoračan, numerički računa Jakobian.
  - Kod *stiff* problema pogodniji od ode45.
- **ode23s** zasnovan na modifikovanoj Rosenbrock formuli drugog reda
  - Jednokoračni, numerički računa Jakobian
  - Efikasniji je od ode15s za manje tačno računanje
  - Bolji za neke *stiff* probleme of ode15s
- **ode23t** zasnovan na trapezoidnom pravilu
  - Pogodan za blage *stiff* probleme
- **ode23tb** je implementacija TR-BDF2: implicitan Runge-Kutta u fazi 1, a trapezoidno pravilo u fazi 2
  - Bolji za neke *stiff* probleme of ode15s

ode23/ode45 – uticaj tolerancije računanja

```
global lambda alpha ncall;      % Globalne promenljive

lambda=1.0; alpha=1.0;          % Model parameteri

for ncase=1:4
   reltol=1.0e-02^(ncase+1);    % tolerancije racunanje
   abstol=reltol;

   ncall=0;                     % reset brojaca

   t0=0.0; tf=10.0;  tout=[t0:1.0:tf]';

   y0=1.0;                      % Inicijana vrednost

   options=odeset('RelTol',reltol,'AbsTol',abstol);
   [t,y]=ode23('ode1p7',tout,y0,options);

   % Prikaz rezultata
   fprintf('reltol = abstol = %6.2e\n\n',reltol);
   fprintf('   t    ye    y    erry\n');

   ye=y0*exp((lambda/alpha)*(1-exp(-alpha*t)));
   erry=ye-y;
   fprintf('%5.1f%9.4f%9.4f%15.10f\n',[t,ye,y,erry]');

   fprintf('\n ncall = %5d\n',ncall);
end
```

```
function yt=ode1p7(t,y)

% globalne promenljive
global lambda alpha ncall;

% ODE
yt=lambda*exp(-alpha*t)*y;

% inkrementiraj brojac
ncall=ncall+1;
```



ode23

```
reltol = abstol = 1.00e-004

   t       ye        y           erry
 0.0   1.0000   1.0000   0.0000000000
 1.0   1.8816   1.8816  -0.0000020034
 2.0   2.3742   2.3743  -0.0000638807
 3.0   2.5863   2.5864  -0.0000933201
 4.0   2.6689   2.6691  -0.0001103656
 5.0   2.7000   2.7002  -0.0001284242
 6.0   2.7116   2.7117  -0.0001434300
 7.0   2.7158   2.7160  -0.0001552112
 8.0   2.7174   2.7175  -0.0001581733
 9.0   2.7179   2.7181  -0.0001592608
10.0   2.7182   2.7183  -0.0001592537
ncall =    73

reltol = abstol = 1.00e-006

   t       ye        y           erry
 0.0   1.0000   1.0000   0.0000000000
 1.0   1.8816   1.8816  -0.0000003221
 2.0   2.3742   2.3742  -0.0000012721
 3.0   2.5863   2.5863  -0.0000016231
 4.0   2.6689   2.6689  -0.0000018847
 5.0   2.7000   2.7000  -0.0000021190
 6.0   2.7116   2.7116  -0.0000023452
 7.0   2.7158   2.7158  -0.0000025176
 8.0   2.7174   2.7174  -0.0000026866
 9.0   2.7179   2.7179  -0.0000028883
10.0   2.7182   2.7182  -0.0000029806
ncall =   268
```

```
reltol = abstol = 1.00e-008

   t       ye        y           erry
 0.0   1.0000   1.0000   0.0000000000
 1.0   1.8816   1.8816  -0.0000000072
 2.0   2.3742   2.3742  -0.0000000188
 3.0   2.5863   2.5863  -0.0000000229
 4.0   2.6689   2.6689  -0.0000000260
 5.0   2.7000   2.7000  -0.0000000285
 6.0   2.7116   2.7116  -0.0000000309
 7.0   2.7158   2.7158  -0.0000000331
 8.0   2.7174   2.7174  -0.0000000354
 9.0   2.7179   2.7179  -0.0000000374
10.0   2.7182   2.7182  -0.0000000394
ncall =  1180

reltol = abstol = 1.00e-010

   t       ye        y           erry
 0.0   1.0000   1.0000   0.0000000000
 1.0   1.8816   1.8816  -0.0000000001
 2.0   2.3742   2.3742  -0.0000000003
 3.0   2.5863   2.5863  -0.0000000003
 4.0   2.6689   2.6689  -0.0000000003
 5.0   2.7000   2.7000  -0.0000000004
 6.0   2.7116   2.7116  -0.0000000004
 7.0   2.7158   2.7158  -0.0000000004
 8.0   2.7174   2.7174  -0.0000000004
 9.0   2.7179   2.7179  -0.0000000005
10.0   2.7182   2.7182  -0.0000000005
ncall =  5392
```

ode45

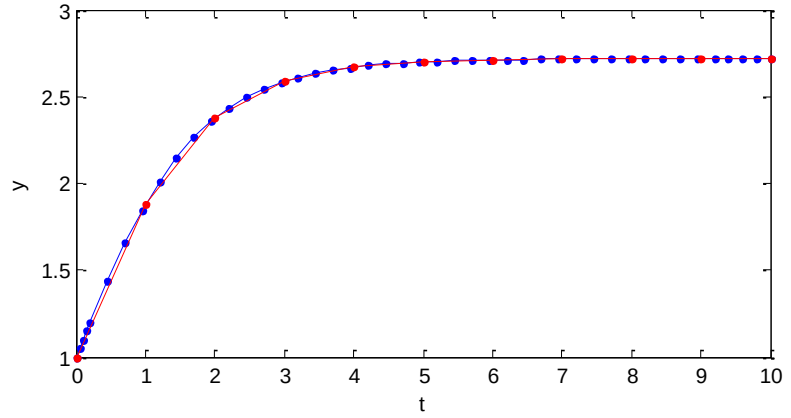
```
reltol = abstol = 1.00e-004

   t       ye        y           erry
 0.0   1.0000   1.0000   0.0000000000
 1.0   1.8816   1.8815   0.0000785756
 2.0   2.3742   2.3742   0.0000230677
 3.0   2.5863   2.5862   0.0000132883
 4.0   2.6689   2.6689   0.0000149230
 5.0   2.7000   2.7000   0.0000165095
 6.0   2.7116   2.7115   0.0000172423
 7.0   2.7158   2.7158   0.0000175329
 8.0   2.7174   2.7174   0.0000176427
 9.0   2.7179   2.7179   0.0000176835
10.0   2.7182   2.7181   0.0000177068
ncall =    73

reltol = abstol = 1.00e-006

   t       ye        y           erry
 0.0   1.0000   1.0000   0.0000000000
 1.0   1.8816   1.8816   0.0000010053
 2.0   2.3742   2.3742   0.0000010986
 3.0   2.5863   2.5863   0.0000011560
 4.0   2.6689   2.6689   0.0000010643
 5.0   2.7000   2.7000   0.0000010083
 6.0   2.7116   2.7116   0.0000009776
 7.0   2.7158   2.7158   0.0000009652
 8.0   2.7174   2.7174   0.0000009604
 9.0   2.7179   2.7179   0.0000009586
10.0   2.7182   2.7182   0.0000009579
ncall =    85
```

```
reltol = abstol = 1.00e-008

   t       ye        y           erry
 0.0   1.0000   1.0000   0.0000000000
 1.0   1.8816   1.8816   0.0000000237
 2.0   2.3742   2.3742   0.0000000012
 3.0   2.5863   2.5863   0.0000000128
 4.0   2.6689   2.6689   0.0000000112
 5.0   2.7000   2.7000   0.0000000428
 6.0   2.7116   2.7116   0.0000000092
 7.0   2.7158   2.7158   0.0000000014
 8.0   2.7174   2.7174  -0.0000000231
 9.0   2.7179   2.7179  -0.0000000081
10.0   2.7182   2.7182   0.0000000077
ncall =   163

reltol = abstol = 1.00e-010

   t       ye        y           erry
 0.0   1.0000   1.0000   0.0000000000
 1.0   1.8816   1.8816   0.0000000001
 2.0   2.3742   2.3742  -0.0000000001
 3.0   2.5863   2.5863  -0.0000000001
 4.0   2.6689   2.6689   0.0000000003
 5.0   2.7000   2.7000   0.0000000004
 6.0   2.7116   2.7116  -0.0000000004
 7.0   2.7158   2.7158   0.0000000003
 8.0   2.7174   2.7174   0.0000000002
 9.0   2.7179   2.7179  -0.0000000003
10.0   2.7182   2.7182   0.0000000001
ncall =   385
```

ode45 – Diferencijalna jednačina 1. reda

```
>> global lambda alpha ncall;
>> lambda=1; alpha=1; ncall=0;
>> tout = [0 10];
>> y0 = 1;
>> [t,y]=ode45('ode1p7',tout,y0);
>> ncall              % broj poziva f-je
ncall =
    67

>> [t y]
ans =
         0    1.0000
    0.0502    1.0502
    0.1005    1.1003
    0.1507    1.1502
    0.2010    1.1997
    0.4510    1.4375
    0.7010    1.6551
    0.9510    1.8472
    1.2010    2.0119
    1.4510    2.1503
    1.7010    2.2648
    1.9510    2.3581
    2.2010    2.4334
    2.4510    2.4937
            ...
    9.8002    2.7181
   10.0000    2.7181

>> size(t,1)          % broj vrsta u t
ans =
    45
```

```
>> ncall=0;
>> tout = 0:10;
>> [t,y]=ode45('ode1p7',tout,y0);

>> ncall
ncall =
    67

>> [t y]
ans =
         0    1.0000
    1.0000    1.8816
    2.0000    2.3742
    3.0000    2.5862
    4.0000    2.6689
    5.0000    2.7000
    6.0000    2.7115
    7.0000    2.7158
    8.0000    2.7173
    9.0000    2.7179
   10.0000    2.7181
```

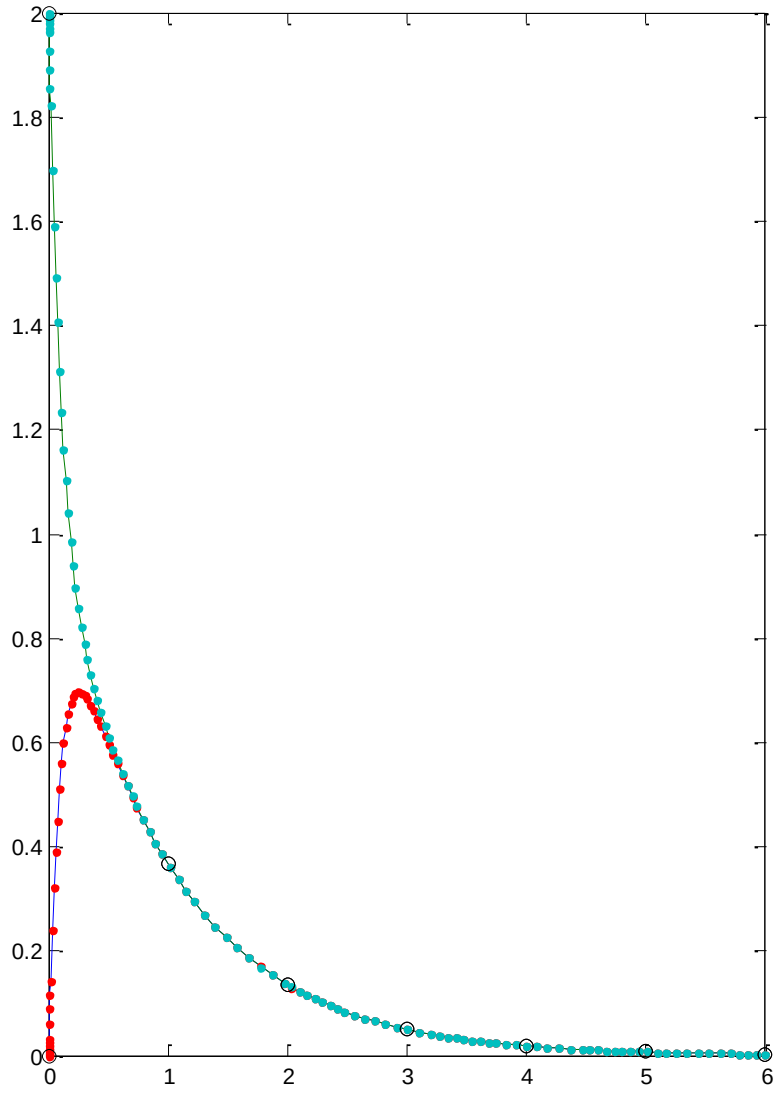


ode45 – Dve diferencijalne jednačine 1. reda = model 2. reda

```
function yt=ode1p8(t,y)
% parametri modela
a=5.5;  b=4.5;
% jednacine
yt(1)=-a*y(1)+b*y(2);
yt(2)= b*y(1)-a*y(2);
yt=yt';
```

```
t0=0.0;  tf=6.0;  tout=[t0 tf];
a=5.5;  b=4.5;
y10=0.0;  y20=2.0;  y0=[y10 y20]';

[t,y]=ode45('ode1p8',tout,y0);

lambda1=-(a-b);  lambda2=-(a+b);
exp1=exp(lambda1*t);  exp2=exp(lambda2*t);
y1e=(y10+y20)/2.0*exp1-(y20-y10)/2.0*exp2;
y2e=(y10+y20)/2.0*exp1+(y20-y10)/2.0*exp2;

[t1,y1]=ode45('ode1p8',t0:tf,y0);

plot(t,[y1e y2e],t,y,'.',t1,y1,'ok');
xlabel('t'), ylabel('y1(t),y2(t)')
```

```
>> size(t)
ans =
   117     1

>> size(y)
ans =
   117     2
```



Lotka–Volterra jednačina

- Model dinamičkog biološkog sistema gde dve vrste imaju odnos: grabljivica-plen

$$\frac{dx}{dt} = x(\alpha - \beta y)$$
$$\frac{dy}{dt} = -y(\gamma - \delta x)$$

Deterministički
Kontinualan
Nelinearan
model 2. reda

x – broj jedinki plena (npr. zečevi)
y – broj jedinki grabljivica (npr. lisice)
izvodi predstavljaju porast broja populacije, *t* vreme, a parametri $\alpha,\beta,\delta,\gamma$ određuju njihovu interakciju.

- Pretpostavke:
  - Plen uvek ima dovoljno hrane
  - Zalihe hrane za grabljivice zavise od populacije plena
  - Stopa promene populacije je proporcionalna njenoj veličini
  - Okruženje se ne menja u korist jedne vrste i nema genetske adaptacije

Promena populacije Plen-Grabljivica

- Promena populacije plena
  - Raste srazmerno broju jedinki zbor reprodukcije
  - Oprada srazmerno učestanosti susreta sa grabljivicama

$$\frac{dx}{dt} = \alpha x - \beta xy$$

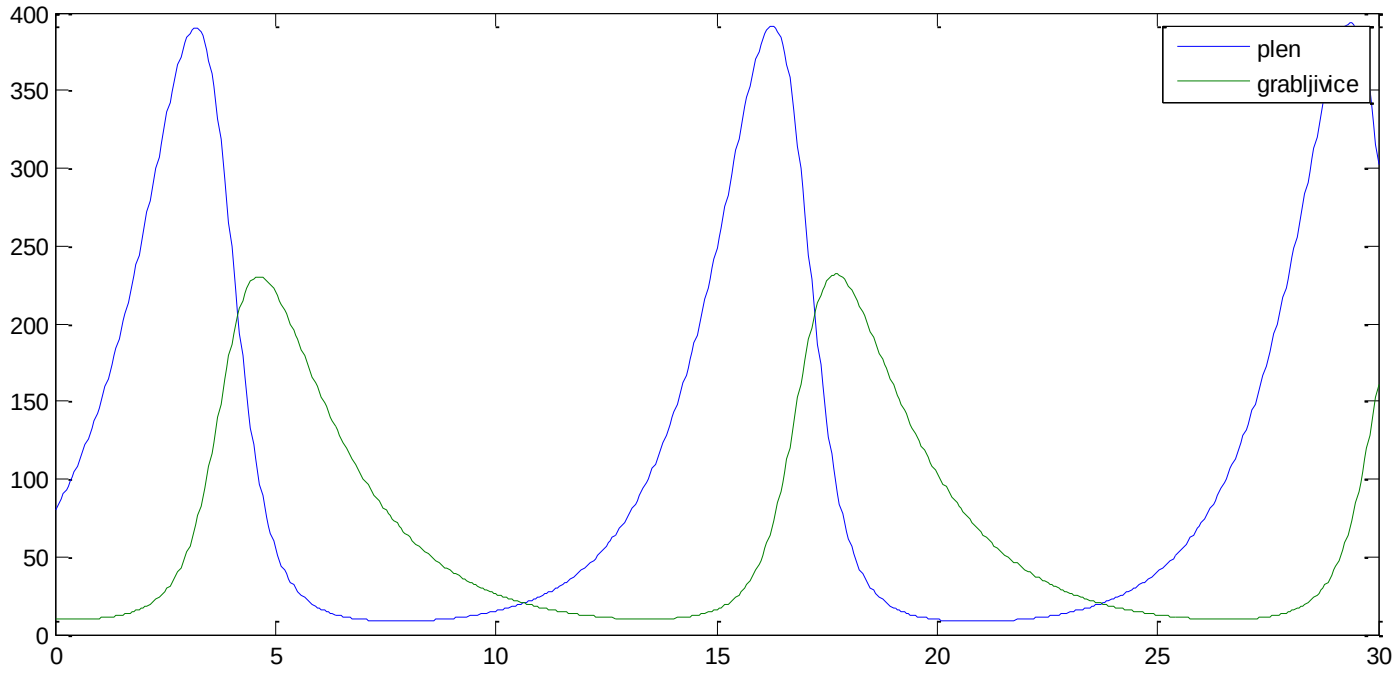
- Promena populacije grabljivica
  - Raste srazmerno ishrani
  - Opada zbog prirodne smrti

$$\frac{dy}{dt} = -\gamma y + \delta xy$$

Simulacija

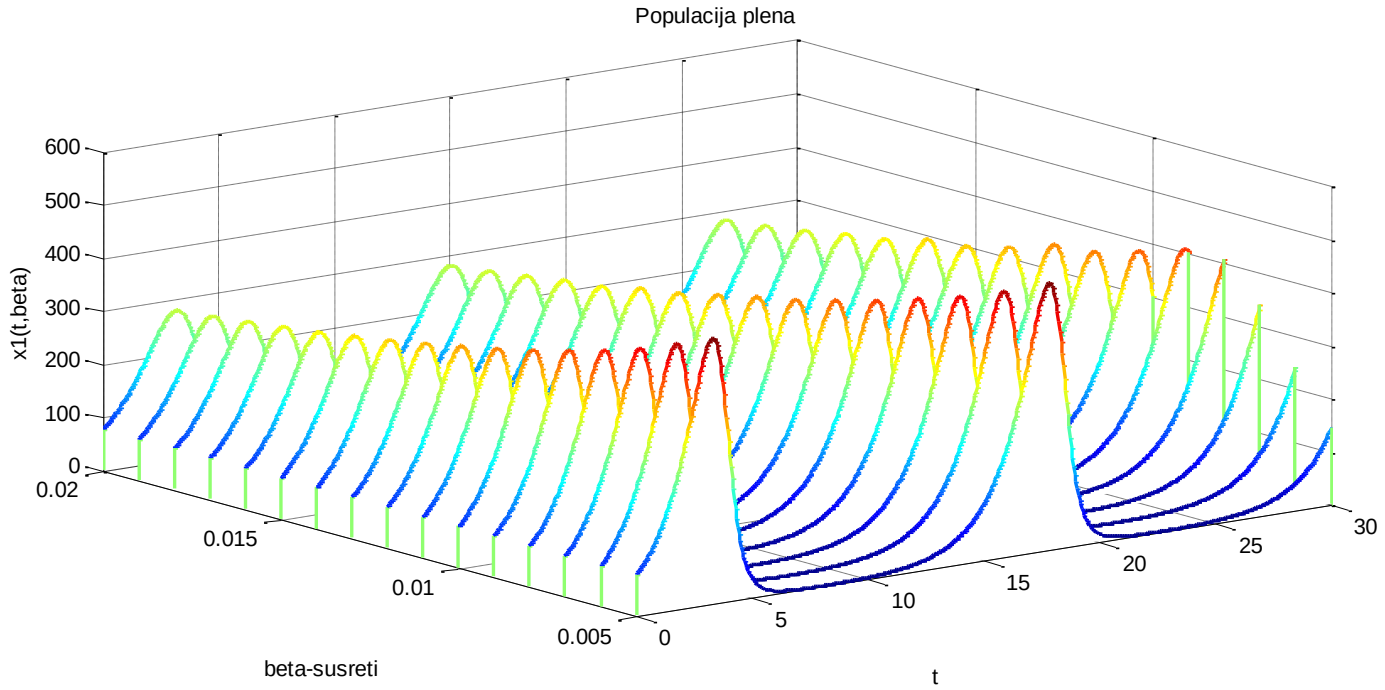
```
function zp = lotka7(t,z,alpha,beta,gama,delta)
% Lotka-Volterra model
x = z(1); y = z(2);
zp = [ x*(alpha - beta*y)
      -y*(gama - delta*x) ];
```

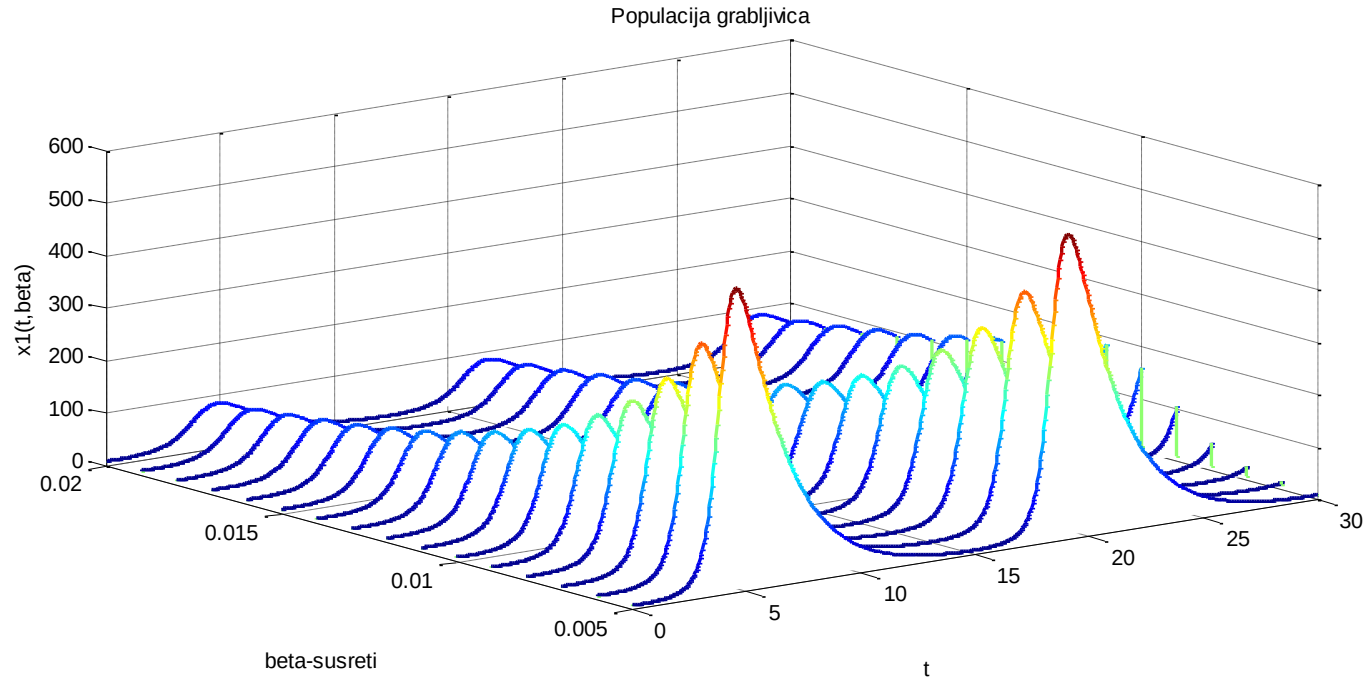
```
al = 0.7; be = 0.01; ga = 0.8; de = 0.05;
tout = 0:0.05:30;
[t,x] = ode45(@lotka7,tout,[40; 80],[], al, be, ga, de);
plot(t,x)
```



Analiza uticaja susreta grabljivica i plena

```
al = 0.7; ga = 0.5; de = 0.005;
beta = 0.005:0.001:0.02;         % parametar susretanja
tout = 0:0.05:30;

xx = []; yy = [];
for be = beta
   [t,x] = ode45(@lotka7, tout, [80; 10], [], al, be, ga, de );
   xx = [xx x(:,1)];
   yy = [yy x(:,2)];
end

[X,Y]=meshgrid(t,beta);
g=waterfall(X,Y,xx');
xlabel('t'), ylabel('beta-susreti'), zlabel('x1(t,beta)')
set(g,'linewidth',2), title('Populacija plena')
pause
g=waterfall(X,Y,yy');
xlabel('t'), ylabel('beta-susreti'), zlabel('x1(t,beta)')
set(g,'linewidth',2), title('Populacija grabljivica')
```


Populacija plena
x1(t,beta)
600
500
400
300
200
100
0
0.02
0.015
0.01
0.005
beta-susreti
0
5
10
15
20
25
30
t


Populacija grabljivica
x1(t,beta)
600
500
400
300
200
100
0
0.02
0.015
0.01
0.005
beta-susreti
0
5
10
15
20
25
30
t

Van der Pol oscillator

- Van der Pol oscilator je nekonzervativni oscilator sa nelinearnim prigušenjem.
- Opisuje se diferencijalnom jednačinom 2. reda:

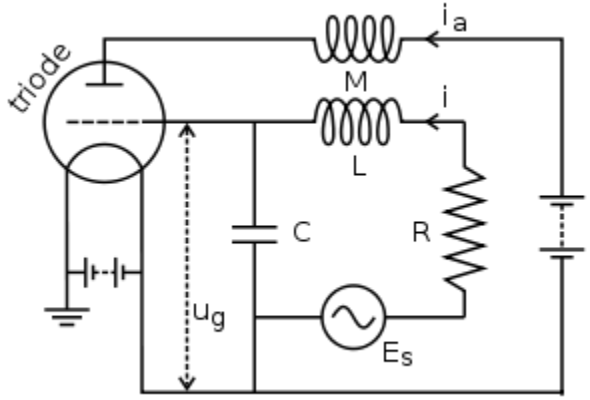
$$\frac{d^2x}{dt^2} - \mu(1-x^2)\frac{dx}{dt} + x = 0$$

gde je *x* promenljiva zavisna od vremena *t*, a μ je parametar (mera nelinearnosti i jačina prigušenja)

- Ovaj oscilator je 1920. godine osmislio holandski el. inženjer i fizičar Balthasar van der Pol dok je radio za Philips.

Balthasar van der Pol

Born	January 27, 1889 Utrecht
Died	October 6, 1959 (aged 70) Wassenaar
Nationality	Dutch
Fields	Physics
Notable awards	IEEE Medal of Honor



Van der Pol jednačina

- Diferencijalna jednačina 2. reda: $\frac{d^2x}{dt^2} - \mu(1-x^2)\frac{dx}{dt} + x = 0$
- Uvodimo smenu: $y = \frac{dx}{dt}$
- Nakon diferenciranja dobijamo: $\frac{dy}{dt} = \frac{d^2x}{dt^2} = \mu(1-x^2)\frac{dx}{dt} - x$

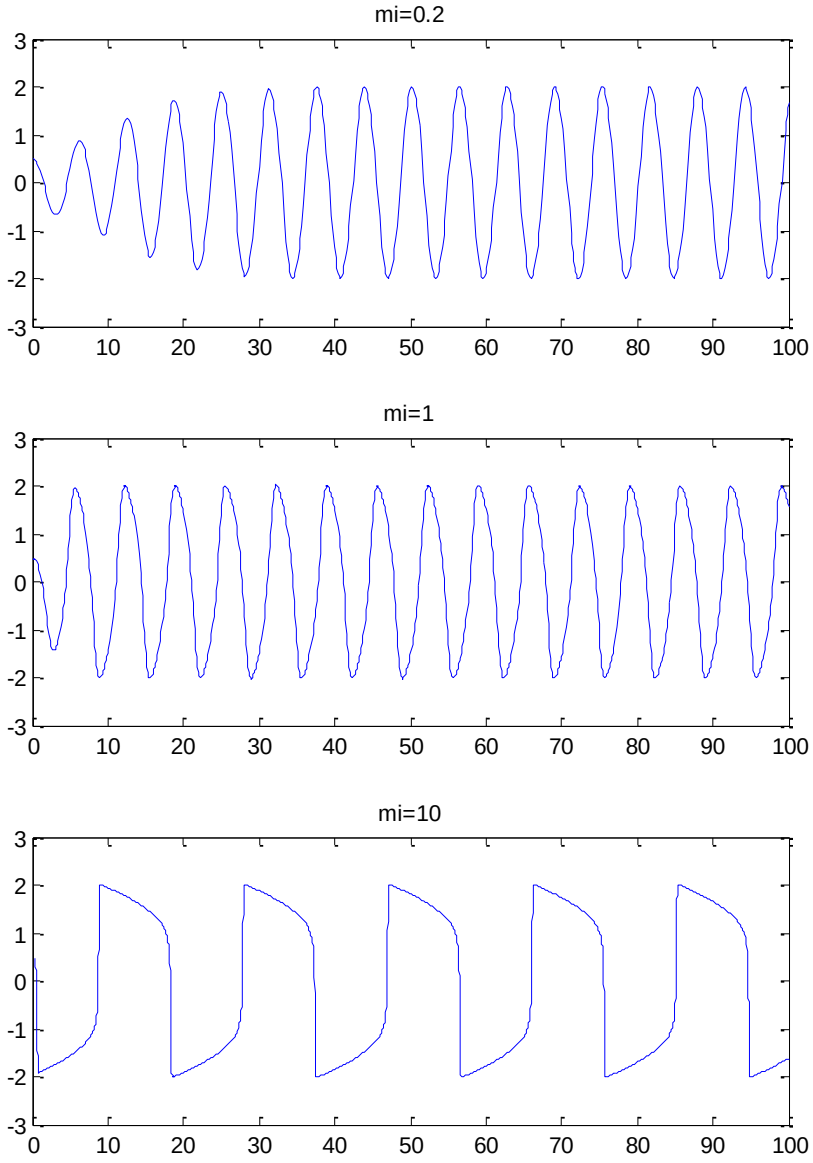
- Konačno model od dve diferencijalne jednačine 1. reda (ekvivalentan polaznoj jednačini)

$$\frac{dx}{dt} = y$$
$$\frac{dy}{dt} = \mu(1-x^2)y - x$$

```
function zprim = vdpol(t, z, mi)
% Opis Van der Pol-ove dif. jednacine
x = z(1);
y = z(2);
xprim = y;
yprim = mi*(1-x^2)*y - x;
zprim = [xprim; yprim];
```

Van der Pol simulacija

```
tp = 0;  tk = 100;  x0 = [0.5 0];

mi = 0.2;
[t, x] = ode45( @vdpol, [tp tk], x0, [], mi);
subplot(3,1,1), plot( t, x(:,1) ),
title('mi=0.2'), axis([0 100 -3 3])

mi = 1;
[t, x] = ode45( @vdpol, [tp tk], x0, [], mi);
subplot(3,1,2), plot( t, x(:,1) ),
title('mi=1'), axis([0 100 -3 3])

mi = 10;
[t, x] = ode45( @vdpol, [tp tk], x0, [], mi);
subplot(3,1,3), plot( t, x(:,1) ),
title('mi=10'), axis([0 100 -3 3])
```



Test solvera

Problem: $m\ddot{x} + c\dot{x} + kx = 0, \quad x(0) = x_0, \quad \dot{x}(0) = v_0$

$x_0 = 1, \quad v_0 = 0$

Nije *stiff* :

$m = 1, \quad c = 11, \quad k = 10$

$x(t) = q_1 e^{s_1 t} + q_2 e^{s_2 t}$

$s_1 = -10, \quad s_2 = -1$

$q_1 = x_0 - \frac{s_1 x_0 - v_0}{s_1 - s_2} = 1 - \frac{10}{9}$

$q_2 = \frac{s_1 x_0 - v_0}{s_1 - s_2} = \frac{10}{9}$

$x(t) = -\frac{1}{9} e^{-10t} + \frac{10}{9} e^{-t}$

Jeste *stiff* :

$m = 1, \quad c = 1001, \quad k = 1000$

$x(t) = q_1 e^{s_1 t} + q_2 e^{s_2 t}$

$s_1 = -1000, \quad s_2 = -1$

$q_1 = x_0 - q_2 = -\frac{1}{999}$

$q_2 = \frac{s_1 x_0 - v_0}{s_1 - s_2} = \frac{1000}{999}$

$x(t) = -\frac{1}{999} e^{-1000t} + \frac{1000}{999} e^{-t}$

Test solvera (2)

Algoritam	Tačaka	Poziva	Traje [ms]		Tačaka	Poziva	Traje [ms]
	Jeste *stiff*				Nije *stiff*		
ode23	4079	12250	502		180	538	35
ode45	12149	19435	732		217	349	15
ode113	6279	13372	900		126	254	19
ode15s	158	194	29		120	150	23
ode23s	260	1298	85		225	1123	75
ode23t	420	469	70		365	401	61
ode23tb	321	808	237		279	591	56

```
global brojac
m=1; k=10; c=11;
opt = odeset('AbsTol',1e-6,'RelTol',1e-6);
brojac=0;
tic;
[t,x]=ode23(@oscil5,[0 10],[1;0],opt,m,c,k);
traje=toc;
```

```
function xp = oscil5(t,x,m,c,k)
global brojac;
xp = [ x(2); (-c*x(2)-k*x(1))/m ];
brojac = brojac + 1;
```