Modelování a simulace - spojité modelování

Slajdy pro předmět IMS

Martin Hrubý

`hrubym @ fit.vutbr.cz`

Vysoké učení technické v Brně

Fakulta informačních technologií,

Božetěchova 2, 61266 Brno

—

prosinec 2005

Půlsemestrální písemka

Spojitá veličina

- hodnota se nemění skokem
- spojitá veličina v počítači

Spojitá simulace

Formulace modelu:

- Diferenciální rovnice
- Schéma s integrátory (blokové)
- Spojité simulační jazyky
- SIMLIB

Pojmy:

- Bloky, Integrátory
- Simulační krok, integrační krok
- Modelový čas

Úvodní příklad

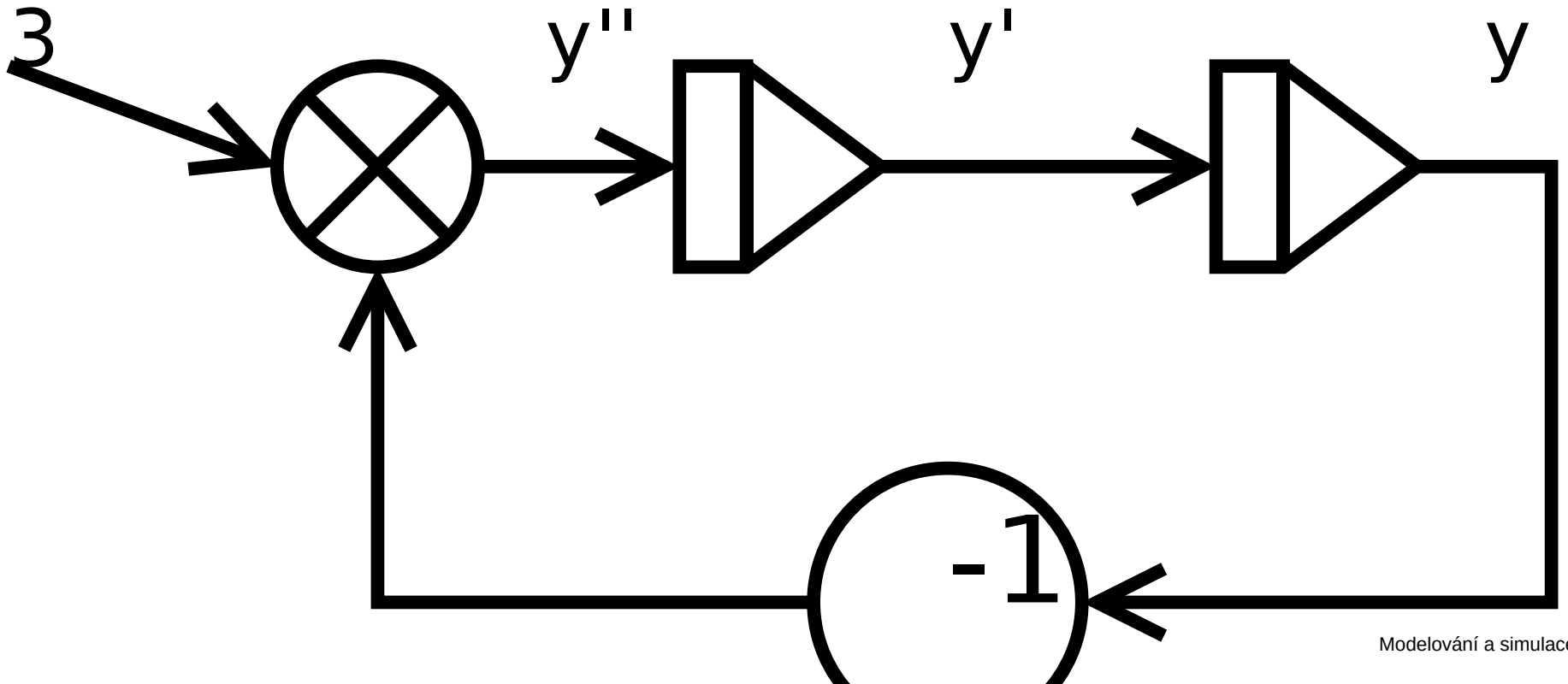
Metoda snižování řádu derivace. Nesmí být derivace vstupů.
Počáteční podmínky.

$y'' + y - 3 = 0$

$y'' = 3 - y$

$y' = \int 3 - y$

$y = \int \int 3 - y$



Simulační model

```
#include "simlib.h"
class   Ex1 {
public:
    Ex1() : y1(3-y), y(y1) {};

    Integrator y1, y;
} EX;
void sample()
{
    Print("%g %g\n", Time, EX.y.Value());
}
Sampler MS(sample, 0.001);
int main()
{
    SetOutput("ex1.dta");
    Init(0, 100);
    Run();
}
```

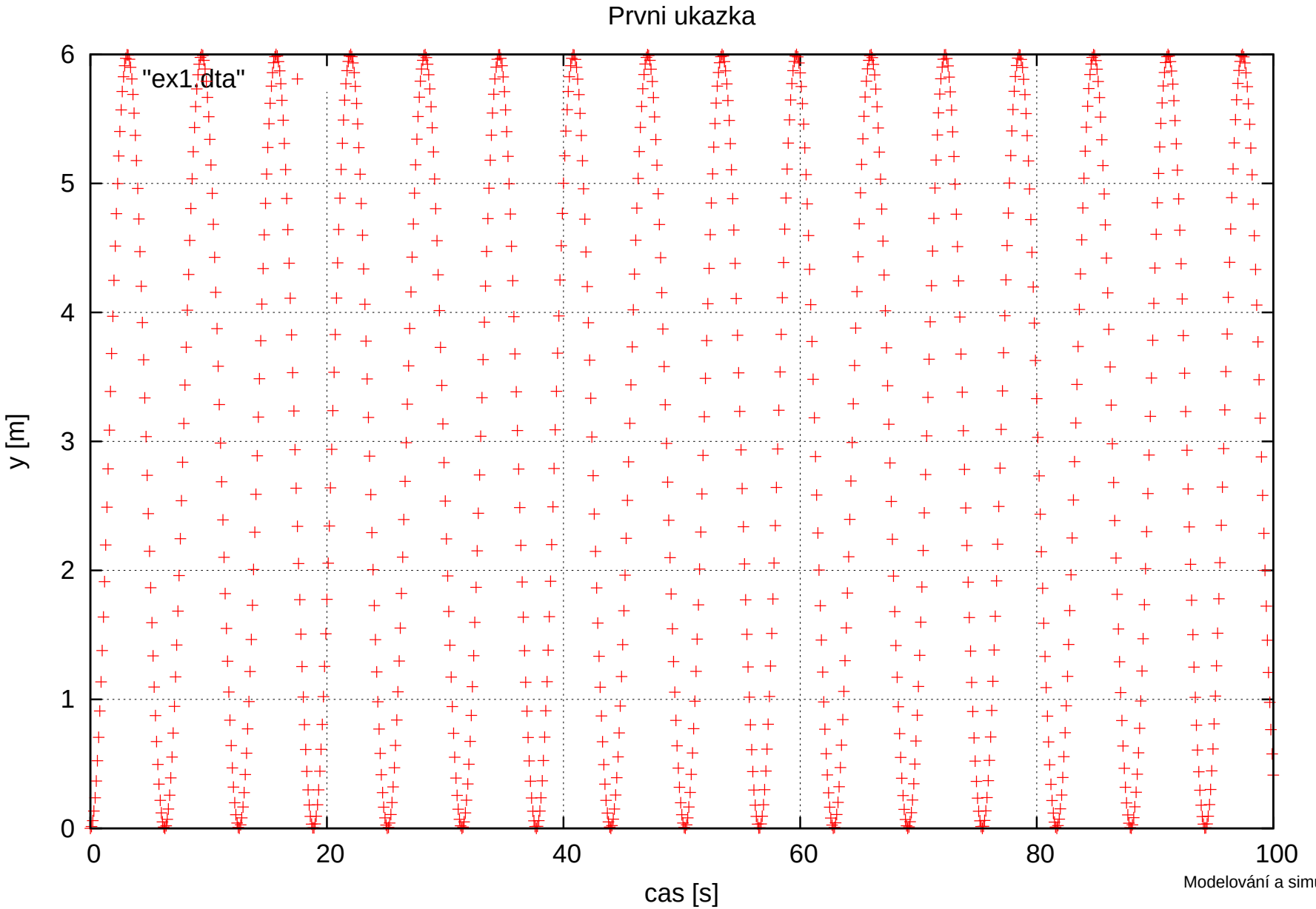
Sampler - implementace

```
class MujSampler : public Process {
    void Behavior() {
       while (1) {
           sample();
           Wait(0.001);
       }
    }
};

class MujSampler2 : public Event {
    void Behavior() {
           sample();
           Activate(Time + 0.001);
    }
};
```

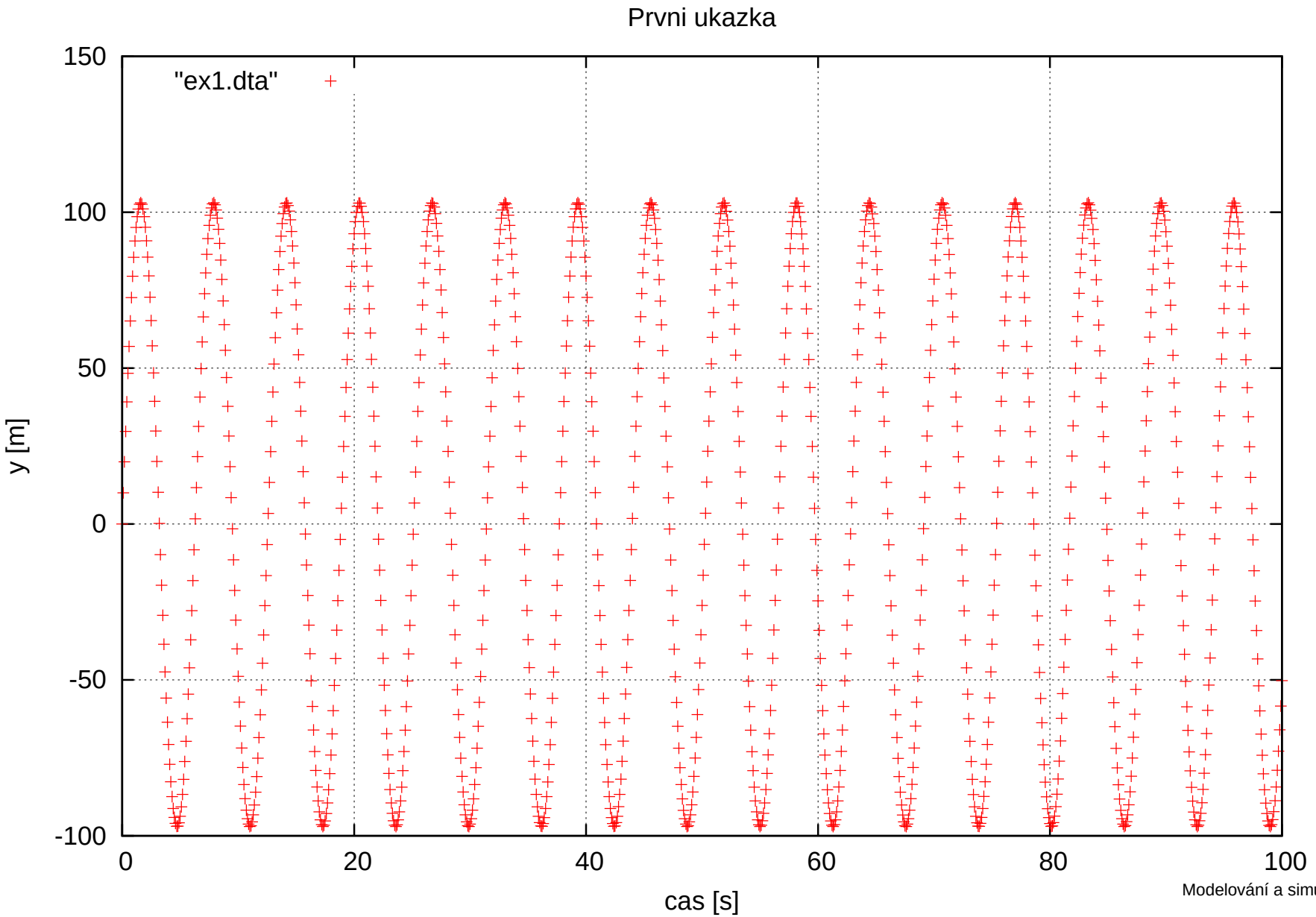
Skript pro gnuplot

```
set term postscript (nebo png)
set output "ex1.eps"
set title "Prvni ukazka"
set grid
set xlabel "cas [s]"
set ylabel "y [m]"
set key left

plot "ex1.dta"
```

Graf



Graf - počáteční podmínky y1(100)



Snižování řádu derivace

$$y^{(5)} + y''' - 8y' + y + 10 = 0$$
$$y^{(5)} = -y''' + 8y' - y - 10$$
$$y^{(4)} = \int (-y''' + 8y' - y - 10)dt$$
$$y^{(3)} = \int \int (-y''' + 8y' - y - 10)dt$$

Snižování řádu derivace

```
class Mujsys {
   // cislo je pocet derivaci na vystupu integratoru
   Integrator y4,y3,y2,y1,y;
   Mujsys() :
        y4(-y3+8*y1-y-1),
        y3(y4),
        y2(y3),
        y1(y2),
        y(y1) {};
};
```

Metoda postupné integrace

vhodná pro rovnice s derivacemi vstupů na pravé straně

1. osamostatnit nejvyšší řád derivace
2. postupná integrace rovnice a
   zavádění nových stavových proměnných
3. výpočet nových počátečních podmínek

Příklad: rovnice $y'' + 2y' + y = x'' + 3x' + 2x$

$p^2y + 2py + y = p^2x + 3px + 2x$

$p^2y = p^2x + p(3x - 2y) + (2x - y)$

$py = px + (3x - 2y) + \frac{1}{p}(2x - y)$, proměnná $w_1 = \frac{1}{p}(2x - y)$

$y = x + \frac{1}{p}(3x - 2y + w_1)$, proměnná $w_2 = \frac{1}{p}(3x - 2y + w_1)$

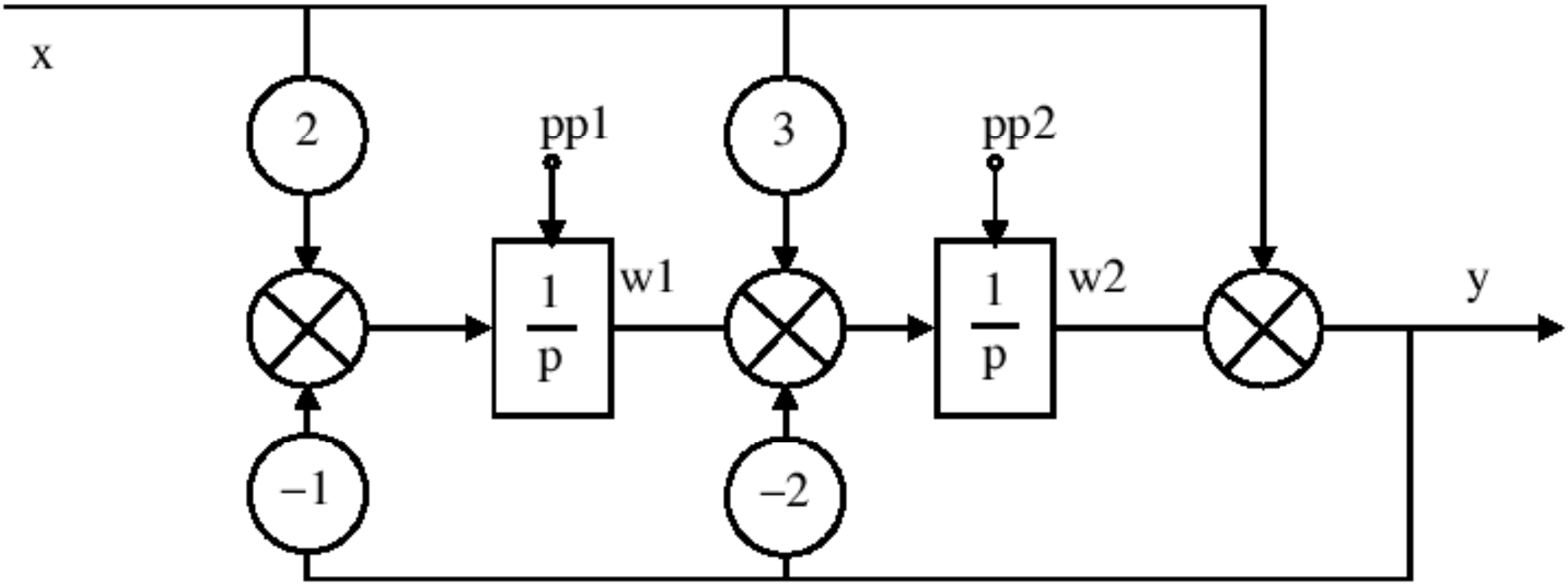
Metoda postupné integrace - příklad

výsledná soustava:

$w_1 = \frac{1}{p}(2x - y)$

$w_2 = \frac{1}{p}(3x - 2y + w_1)$

$y = x + w_2$



Eulerova metoda – princip

$$y(t+h) = y(t) + hf(t, y(t))$$

Eulerova metoda – implementace

```
double  yin[2],    y[2] = { 1.0, 0.0 };
double  time = 0,  h = 0.001;
void update() {    // výpočet vstupů integrátorů
    yin[0] = y[1];      // y'
    yin[1] = -y[0];     // y''
}
void integrate_euler() { // krok integrace
    update();
    for (int i = 0; i < 2; i++)
        y[i] += h * yin[i];
    time += h;
}
int main() {
    while (time < 20) {
        printf("%10f %10f\n", time, y[0]);
        integrate_euler();
    }
    return 0;
}
```

Řízení spojité simulace

```
public SpojitaSimulace : public Process {

     void Behavior() {
          while (1) {
               spocti_vstupy_integratoru();
               udelej_integracni_krok();
               Wait(STEP);
          }
     }
}
```

Vícekrokové metody

- Adams-Bashforth:

$$y_{n+1} = y_n + \frac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})$$

- Metody typu prediktor/korektor zpřesňují výsledek Adams-Bashforth-Moulton:

$$y_{n+1} = y_n + \frac{h}{24}(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2})$$

Poznámka: Problém startu metody – použití jednokrokové metody.

Van der Pol oscilator

Ukázka nelineárního systému.

$x'' + x = a(1 - x^2)x'$

a je parametr

$x'' = a(1 - x^2)x' - x$

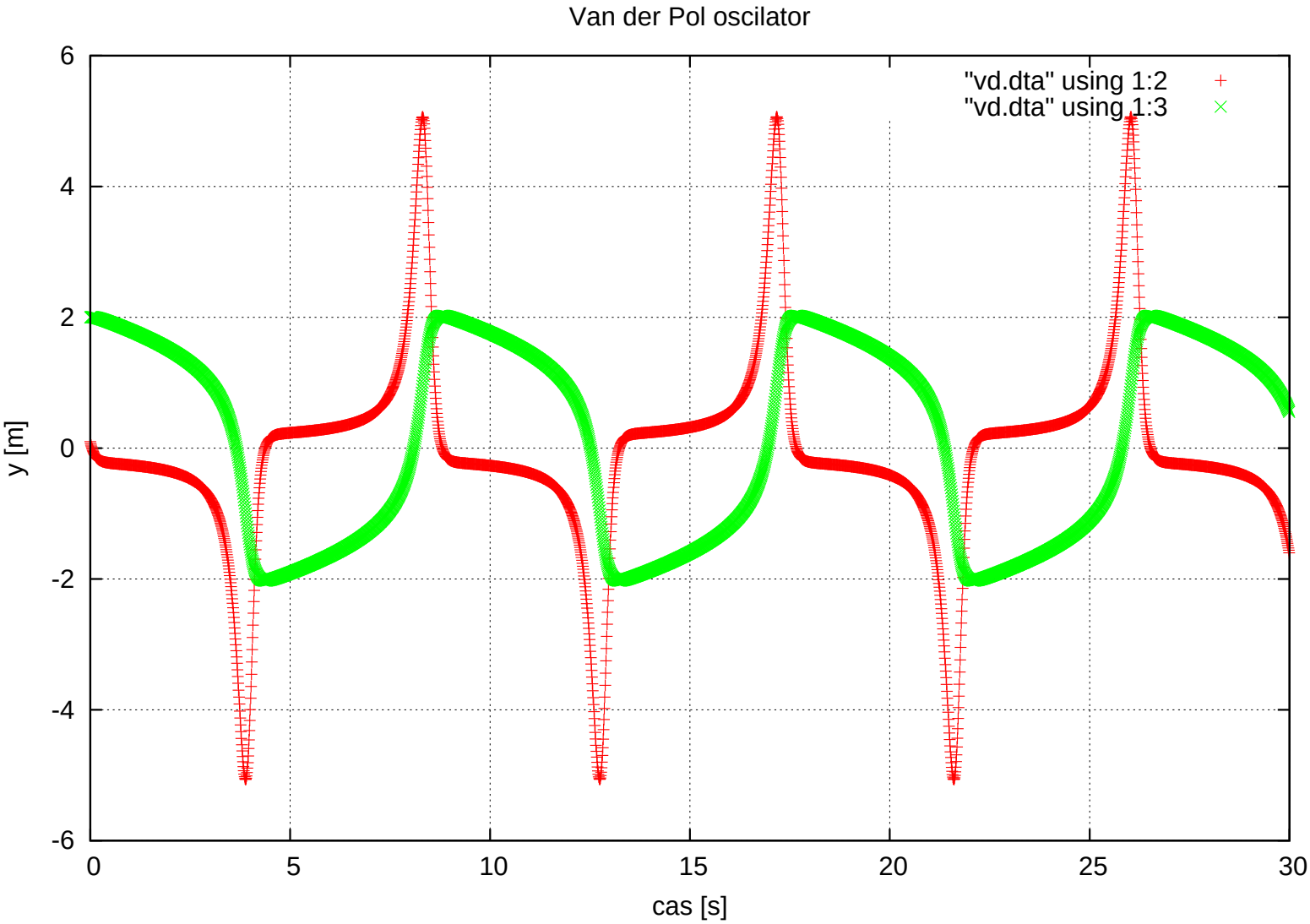
Sim. model

```
const double px1 = 0.1;
const double px = 2;

class   Vd {
public:
    Vd() : a(1),
      x1(a*(1-x*x)*x1 - x, px1),
      x(x1,px){};

    Integrator x,x1;
    Parameter a;
} Vd;

void sample()
{
    Print("%g %g %g\n", Time, Vd.x1.Value(), Vd.x.Value()
}
```

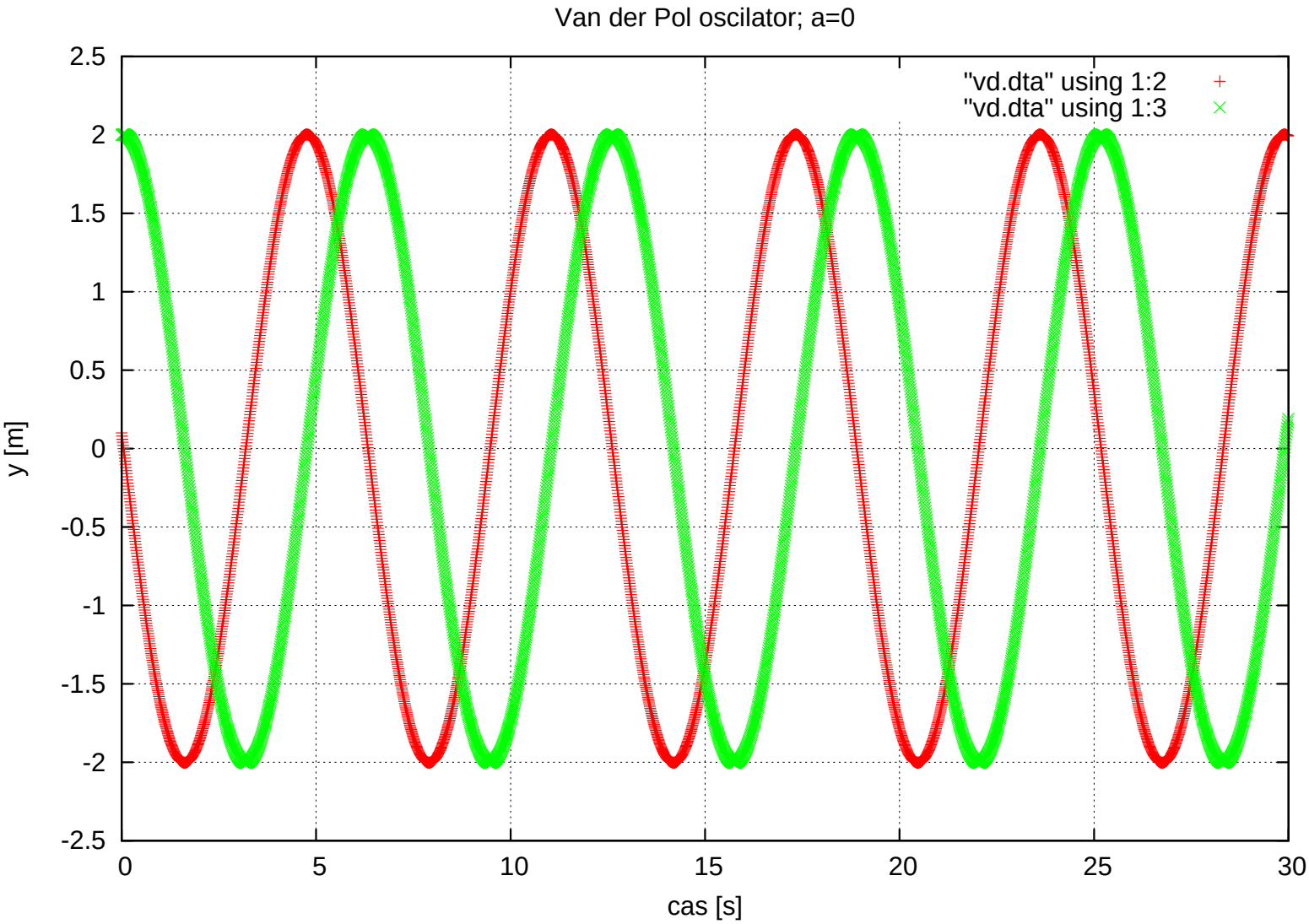
Sim. model

```
Sampler MS(sample, 0.1);

int main(int argc, char *argv[])
{
    Vd.a = atof(argv[1]);
    SetStep(0.001, 0.1);
    SetOutput("vd.dta");
    Init(0, 30);
    Run();
}
```

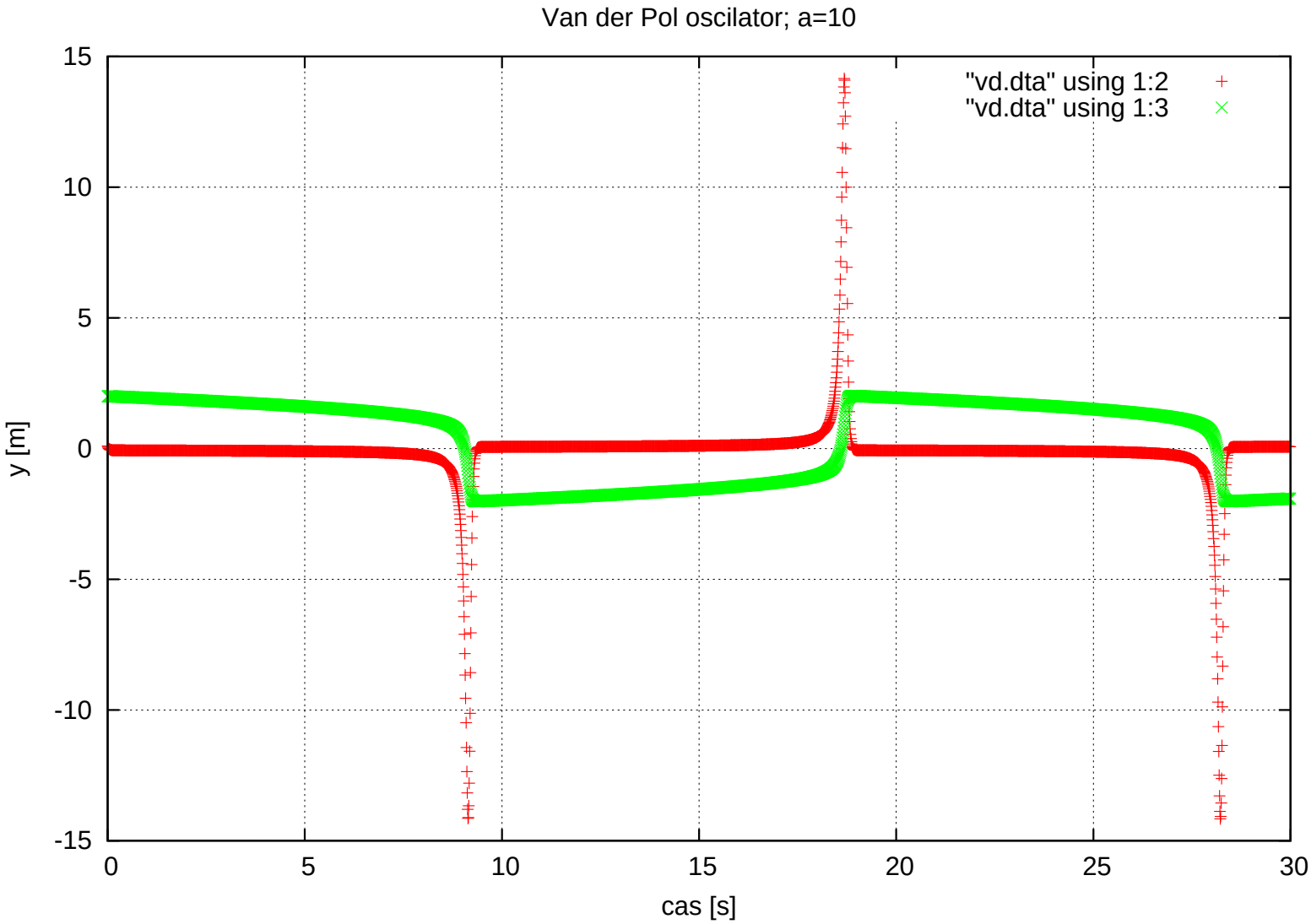
Výsledky


Van der Pol oscilator
"vd.dta" using 1:2
"vd.dta" using 1:3
y [m]
cas [s]
6
4
2
0
-2
-4
-6
0
5
10
15
20
25
30

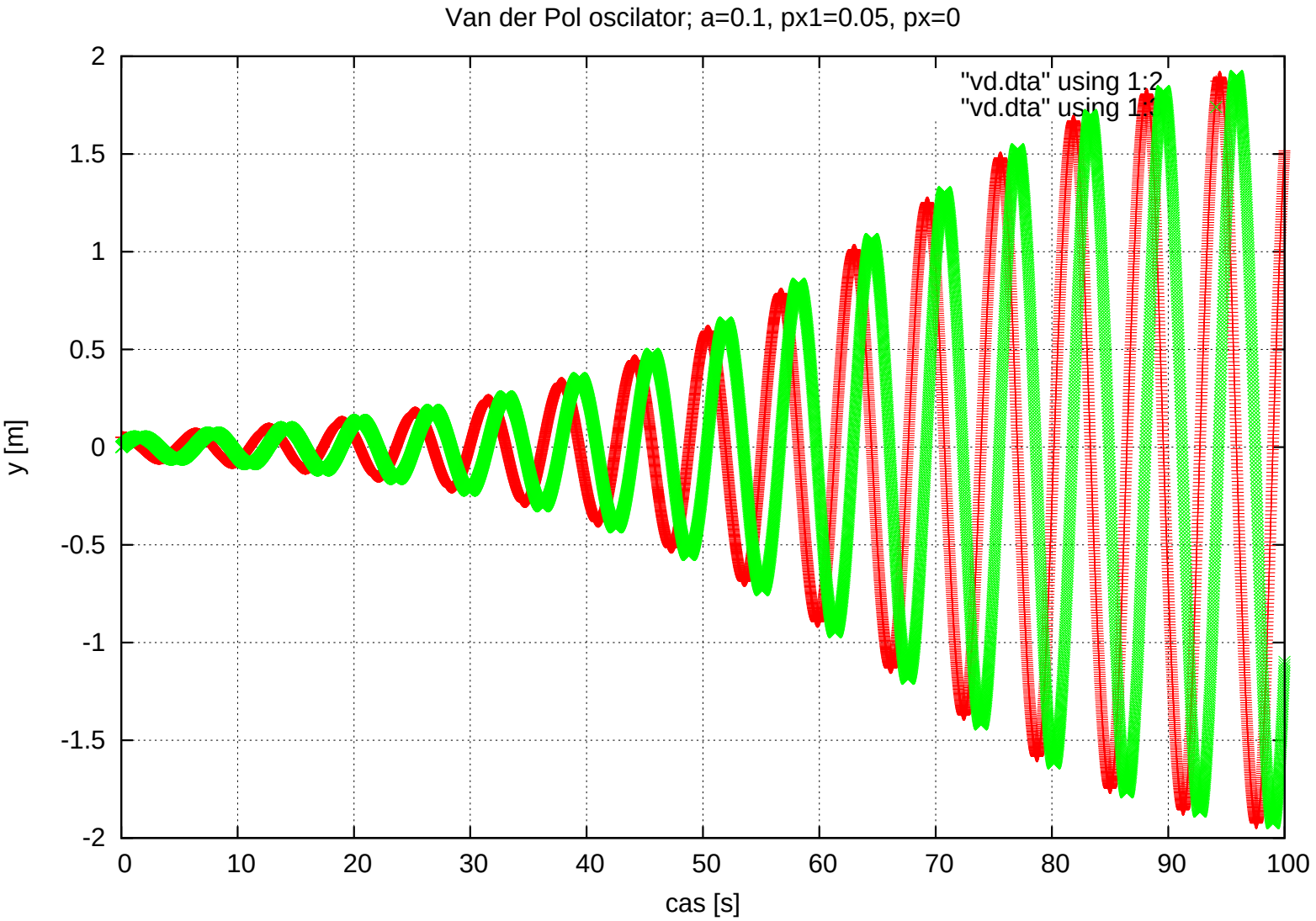
Výsledky


Van der Pol oscilator; a=0
"vd.dta" using 1:2
"vd.dta" using 1:3
y [m]
cas [s]
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
5
10
15
20
25
30

Výsledky

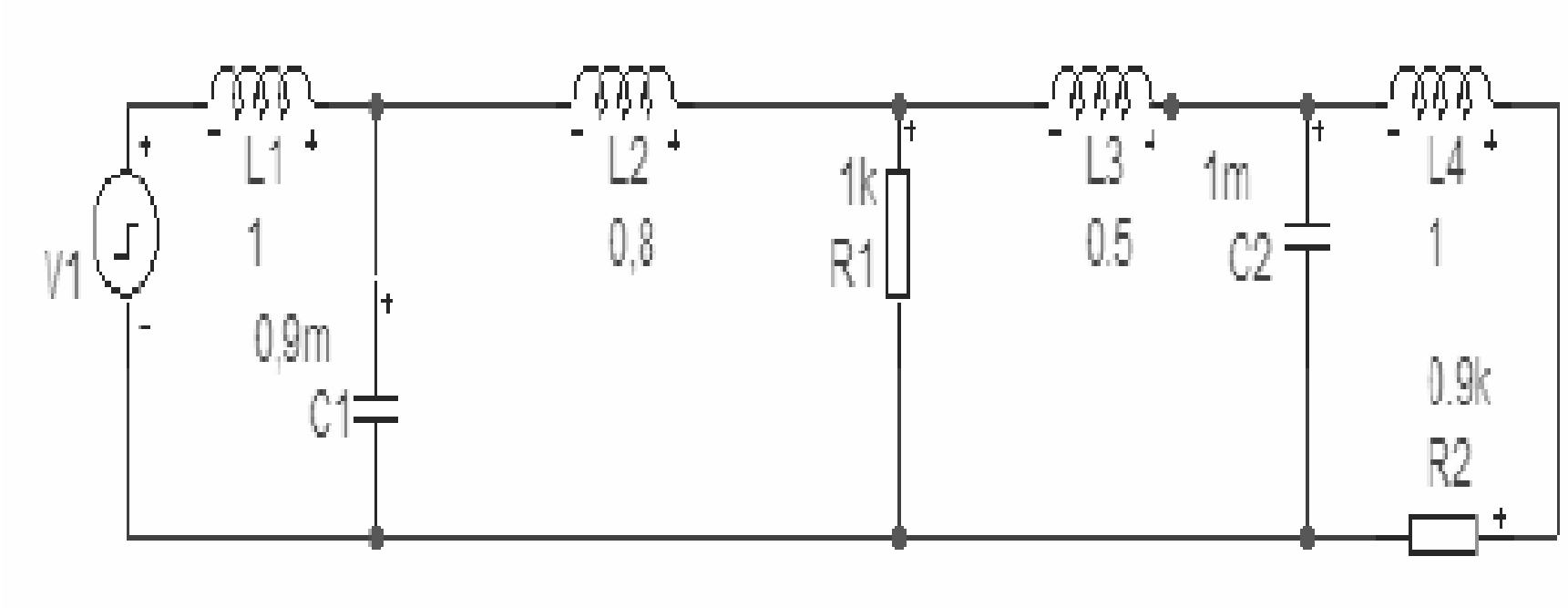

Van der Pol oscilator; a=10
"vd.dta" using 1:2
"vd.dta" using 1:3
y [m]
cas [s]
15
10
5
0
-5
-10
-15
0
5
10
15
20
25
30

Výsledky


Van der Pol oscilator; a=0.1, px1=0.05, px=0
"vd.dta" using 1:2
"vd.dta" using 1:3
y [m]
cas [s]
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
10
20
30
40
50
60
70
80
90
100

Modely obvodů

$$u_L = Li'$$
$$u_C = \frac{1}{C}\int i\ dt$$



(projekt MSI, Josef Lukaštík)

Rovnice obvodu

Smyčka 1: $L_1 i_1' + \frac{1}{C_1}\int(i_1 - i_2)dt - U = 0$

$x = \frac{1}{C_1}\int(i_1 - i_2)dt$

$x' = \frac{1}{C_1}(i_1 - i_2)$

$L_1 i_1' + x = U$

$i_1' = \frac{U-x}{L_1}$

Smyčka 2: $L_2 i_2' + \frac{1}{C_1}\int(i_2 - i_1)dt + R_1(i_2 - i_3) = 0$

$L_2 i_2' - x + R_1(i_2 - i_3) = 0$

$i_2' = \frac{x - R_1(i_2 - i_3)}{L_2}$

Smyčka 3:

$L_3 i_3' + \frac{1}{C_2}\int(i_3 - i_4)dt + R_1(i_3 - i_2) = 0$

$y = \frac{1}{C_2}\int(i_3 - i_4)dt$

$y' = \frac{1}{C_2}(i_3 - i_4)dt$

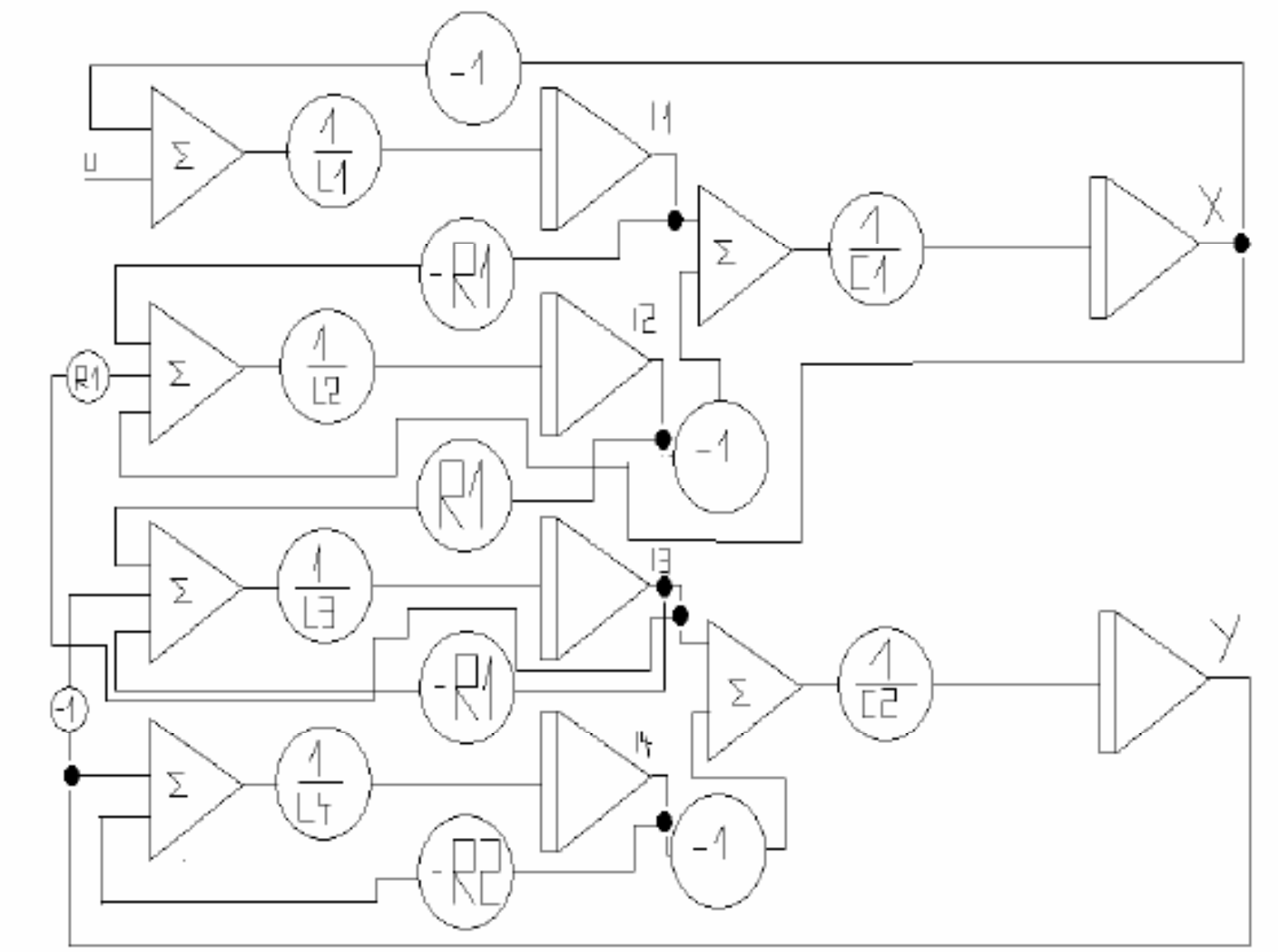
$L_3 i_3' + y + R_1(i_3 - i_2) = 0$

$i_3' = \frac{-y - R_1(i_3 - i_2)}{L_3}$

..rovnice

$$L_4 i_4' + \frac{1}{C_2}\int (i_4 - i_3)dt + R_2 i_4 = 0$$
$$L_4 i_4' - y + R_2 i_4 = 0$$
$$i_4' = \frac{y - R_2 i_4}{L_4}$$

Schema s integrátory



Obvod: SM

```
#include "simlib.h"

// hodnoty jednotlivych prvku

double U=50;
double C1=0.0009;
double C2=0.001;
double L1=1;
double L2=0.8;
double L3=0.5;
double L4=0.5;
double R1=1000;
double R2=900;
```

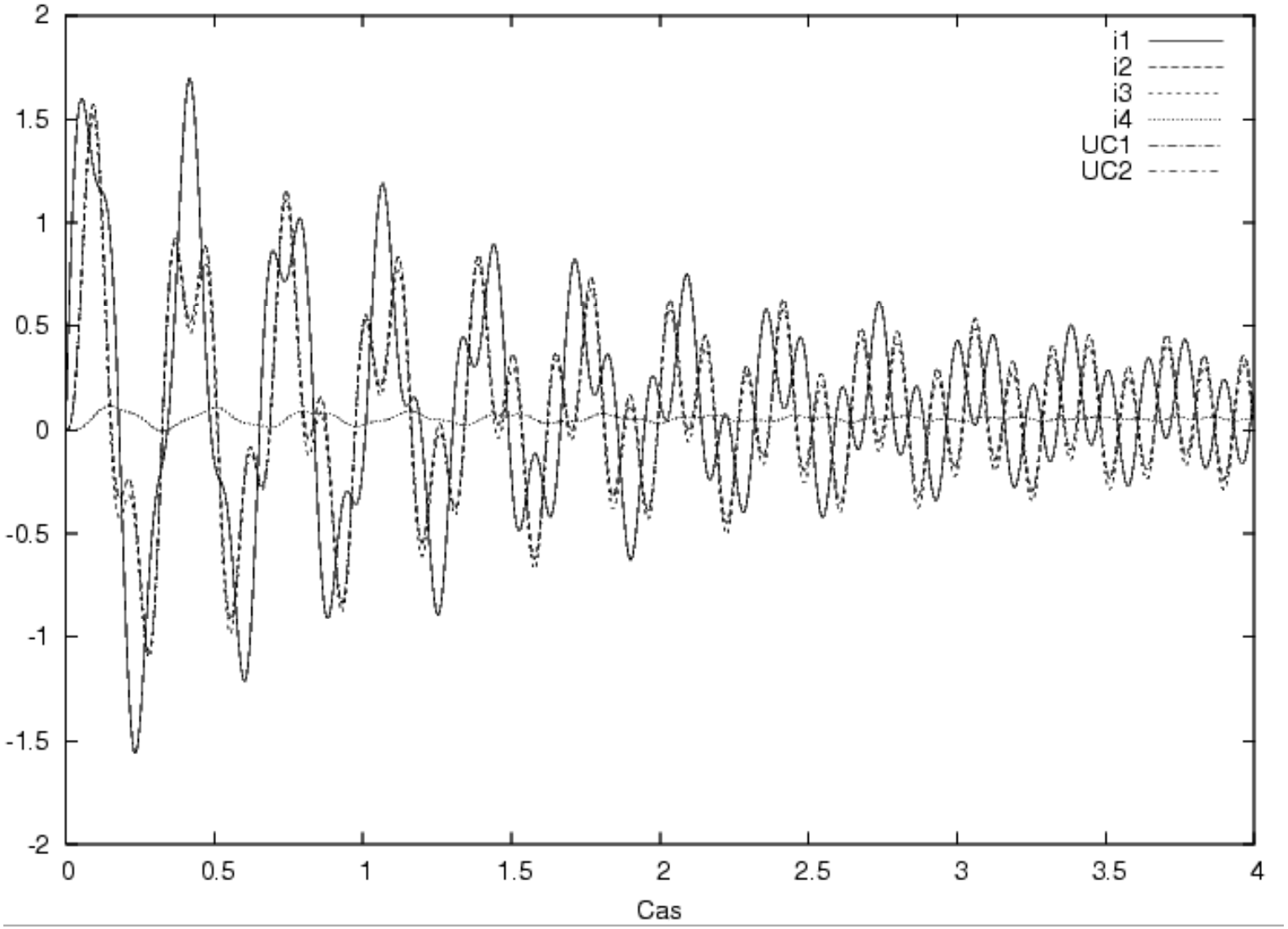
Obvod: SM

```
class Schema {
  Integrator X, Y, I1, I2, I3, I4;
  Schema():
    X((I1-I2)/C1),
    Y((I3-I4)/C2),
    I1((U-X)/L1),
    I2((X-R1*(I2-I3))/L2),
    I3((-Y-R1*(I3-I2))/L3),
    I4((Y-R2*(I4))/L4){}
  double UC1() { return X.Value();}
  double UC2() { return Y.Value();}
  double II1() { return I1.Value();}
  double II2() { return I2.Value();}
  double II3() { return I3.Value();}
  double II4() { return I4.Value();}
};
```

Obvod: SM

```
Schema o;

void Sample() {                    // vzorkovací funkce
  Print("%-8g\t%g\t%g\t%g\t%g\n", T.Value(),o.II1(), o.I
//  Print("%-8g\t%g\t%g\n", T.Value(),o.UC1(), o.UC2());
}
Sampler S(Sample,0.001);           // vzorkovací objekt

int main() {                       // popis experimentu
  SetOutput("vystup.dat");
  SetStep(0.00001,0.0001);          // krok
  Init(0,4);                       // inicializace experimen
  Run();                           // simulace
  return 0;
}
```

Výsledek



Kombinovaný systém

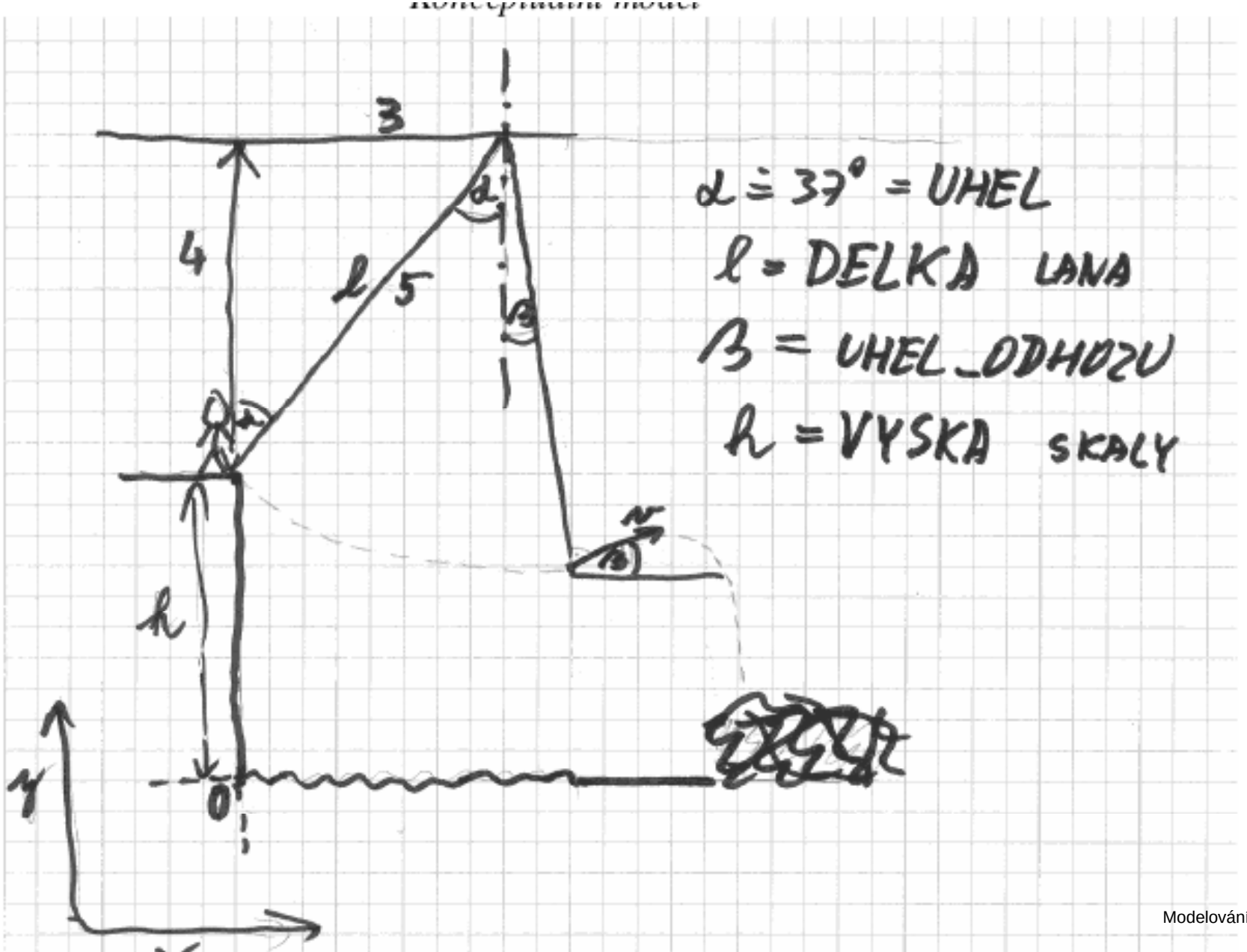
Jednoho dne se Tarzan procházel po pralese, když tu najednou zaslechl volání o pomoc. Pochopil, že krásná Jane je asi v nesnázích, a tak se hned rozběhl směrem, odkud volání přicházelo. Dorazil až na okraj skály tyčící se nad divokou řekou. Protože ale Jane miloval, nemohla jej tato překážka zastavit. Nad řeku se skláněl starý strom, z něhož visely liány. Jednu z nich si dlouhým klackem přitáhl a chtěl se zhoupnout na druhý břeh. Jenže kousek od břehu bylo mnoho velmi ostrých kamenů... Kdy se má Tarzan liány pustit, aby se neutopil ani nepořezal o kameny?

Tarzan stojí na skále ve výšce 4m nad hladinou řeky. Z větve stromu (4m nad ním) sklánějícího se nad řeku si přitáhne liánu, která je od něj vzdálená 3m. Druhý břeh je v úrovni hladiny.

Řeka je široká 4m, 2m od ní leží oblast posetá ostrými kameny.

(MSI projekt, Jan Zezulka)

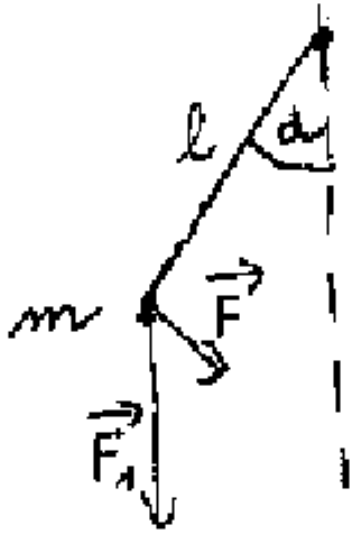
Konceptuální model



Abstraktní modely

Abstraktní modely

KYVADLO



$y = l \cdot (1 - \cos\alpha)$

$x = l \cdot \sin\alpha \doteq l \cdot \alpha$

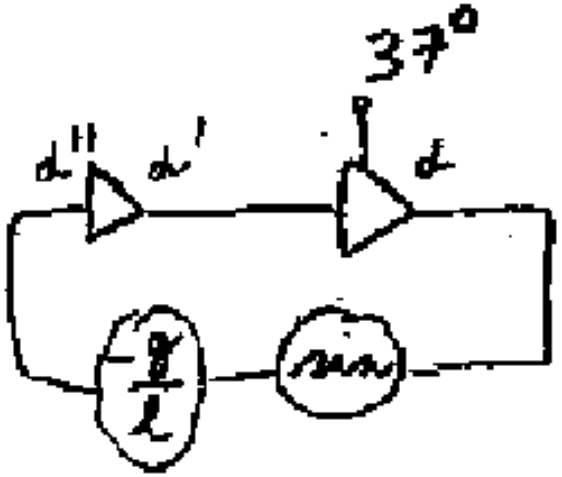
$F = F_1$

$m \cdot x'' = -m \cdot g \cdot \sin\alpha$

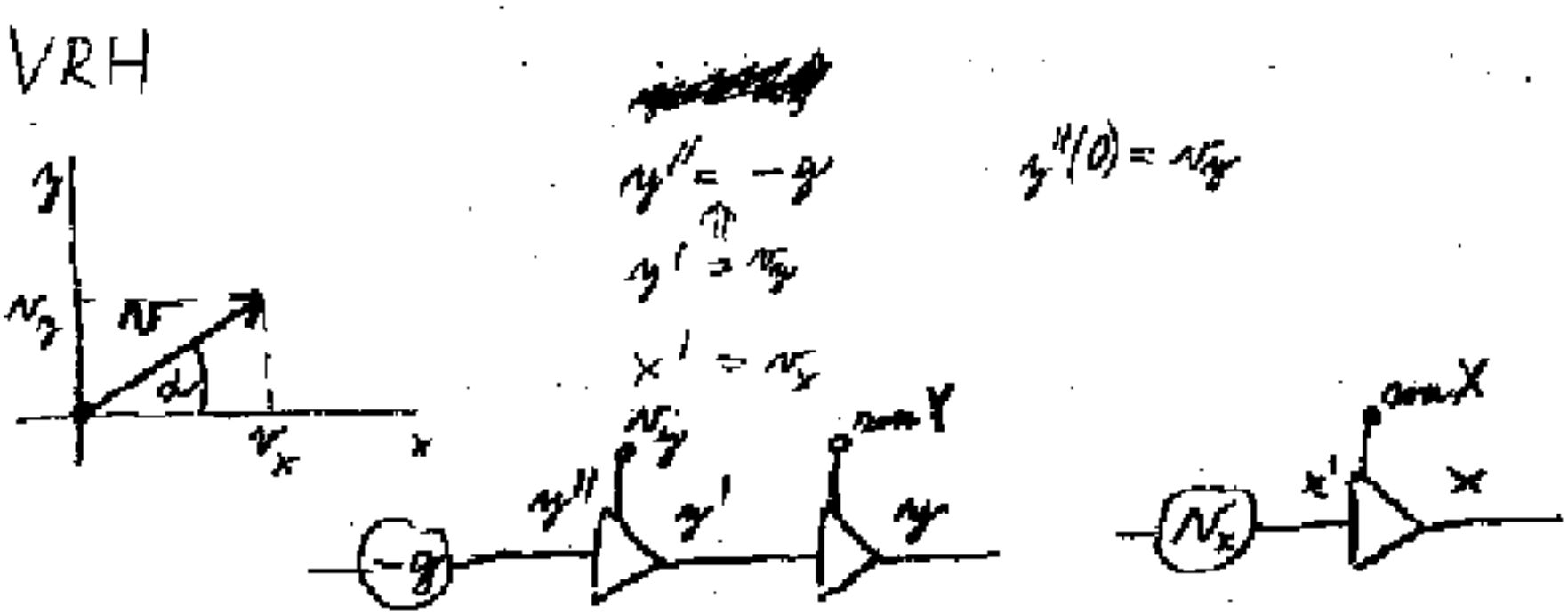
$x'' = -g \cdot \sin\alpha$

$\Downarrow$

$\alpha'' = -\frac{g}{l} \cdot \sin\alpha$



Abstraktní modely



Simulační modely

```
class Kyvadlo{

Integrator alfa, dalfa;
double x, y, last_x, last_y;  //souradnice

Kyvadlo():
dalfa((-g/DELKA)*Sin(alfa)),
        alfa(dalfa, UHEL),
        x(0.0), y(VYSKA), last_x(0.0), last_y(VYSKA) {}

};
Kyvadlo *k;
```

Simulační modely

```
void Kyvadlo::Poloha() {
  if (alfa.Value() < uhel_odhozu) {
    x =(sin(alfa.Value()*Pir) * DELKA) - (sin(UHEL*Pir) *
    y =DELKA * (1-cos(alfa.Value()*Pir)) + (VYSKA+DELKA*(
    vx = (x-last_x)/SAMPLE_STEP;
    vy = (y-last_y)/SAMPLE_STEP;
    last_x = x;
    last_y = y;
    //Print("%g ", alfa.Value());
    souX = x;
    souY = y;
    beta = alfa.Value();
  } else {                    //ukonceni kyvadla
    cas = T.Value();
    beta = alfa.Value();
    Stop();
  }
 }
```

Simulační modely

```
class Vrh:ConditionDown {
public:
Integrator dy, y, x;
Vrh():ConditionDown(y),
       dy(-g, vy),
       y(dy, souY),
       x(vx, souX) {}
void Action() {
  //Print("konec");
  Stop();}
};
Vrh *r;
```

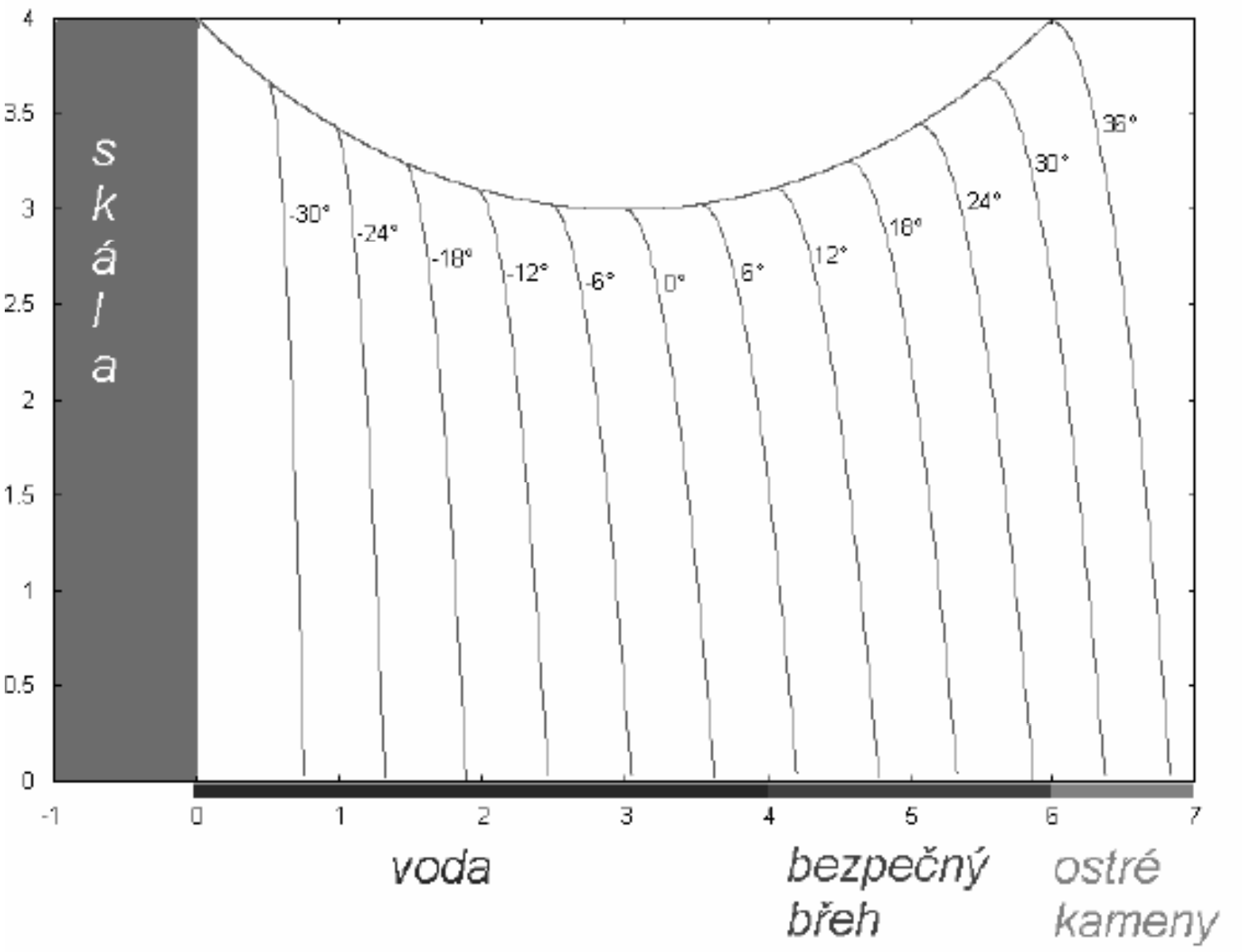
Simulační modely

```
void Sample(){
  if (beta < uhel_odhozu) {
    k->Poloha();
    Print("%g %g %g\n", T.Value(), k->x, k->y);
  } else
    Print("%g %g %g\n", T.Value(), r->x.Value(), r->y.Va
}
Sampler S(Sample,SAMPLE_STEP);
```

Simulační modely

```
int main()
{
  SetOutput("msiproj.dat");    //presmerovani vystupu
  for(uhel_odhozu=-36.0; uhel_odhozu<=36.0; uhel_odhozu+=
  //simulujeme pro ruzne uhly opusteni lana
    Kyvadlo K; k=&K;
    Init(0, 30);               //nastaveni casu
    SetStep(1e-3, 0.1);        //nastaveni kroku
    Run();
    //Print("tady");
    Vrh R; r=&R;
    Init(cas, 60);             //nastaveni casu (posunute o
    SetStep(1e-3, 0.1);        //nastaveni kroku
    Run();
    Print("\n");
  }
  return 0;
}
```

Výsledky


4
3.5
3
2.5
2
1.5
1
0.5
0
-1
0
1
2
3
4
5
6
7
skála
-30°
-24°
-18°
-12°
-6°
0°
6°
12°
18°
24°
30°
36°
voda
bezpečný břeh
ostré kameny

Simulační modely