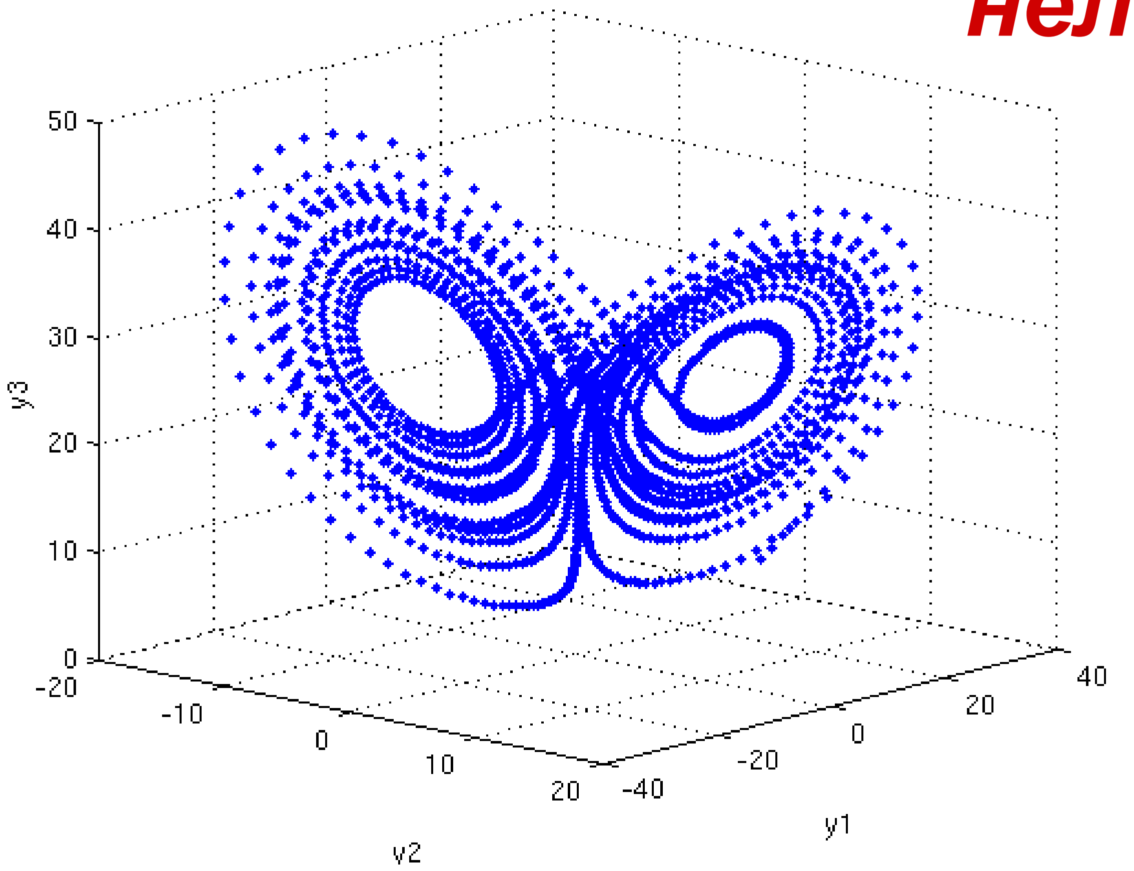
Математическое моделирование РТУ и С

Лекция 9. Моделирование нелинейных звеньев



Преподаватель:

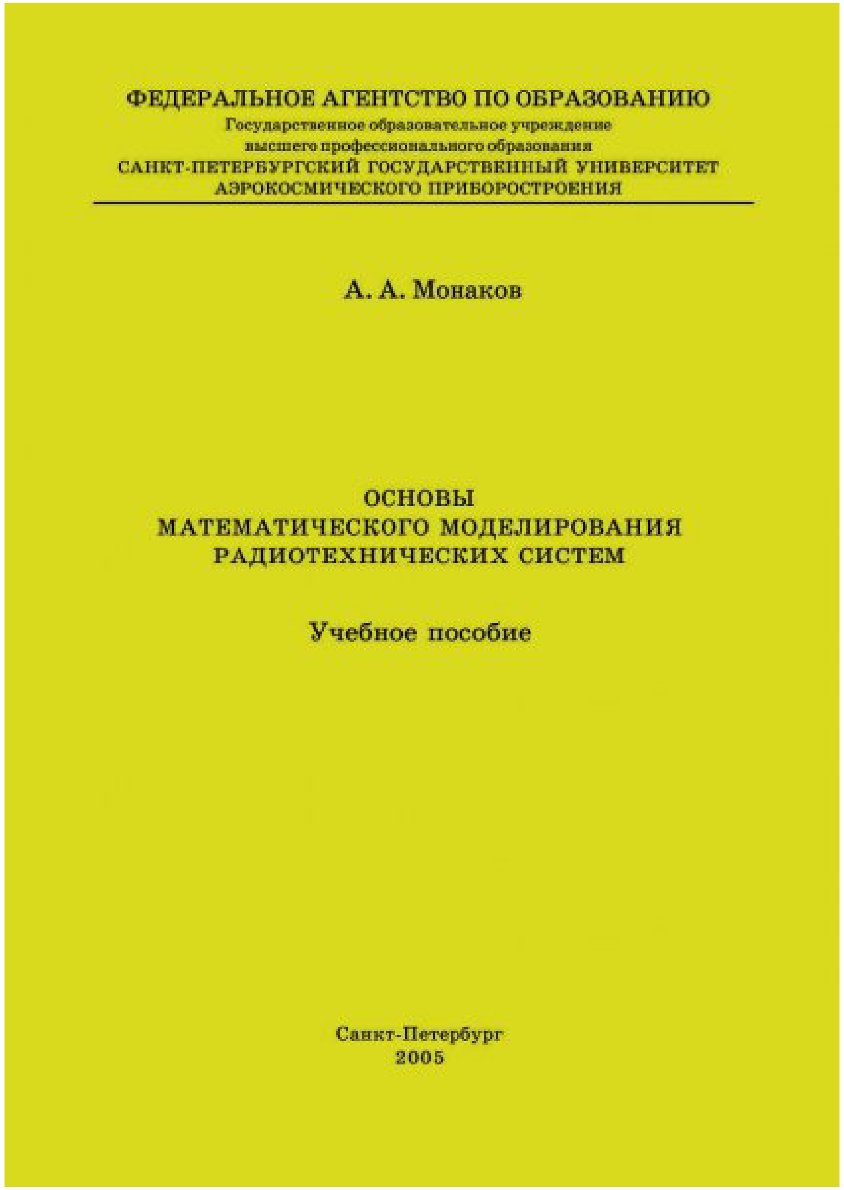
Корогодин Илья

korogodin@srns.ru

Литература

Монаков А.А. Основы математического моделирования радиотехнических систем. Учебное пособие. – СПб.: ГУАП, 2005. – 100с.

Глава 2, раздел 2.2. Моделирование нелинейных систем

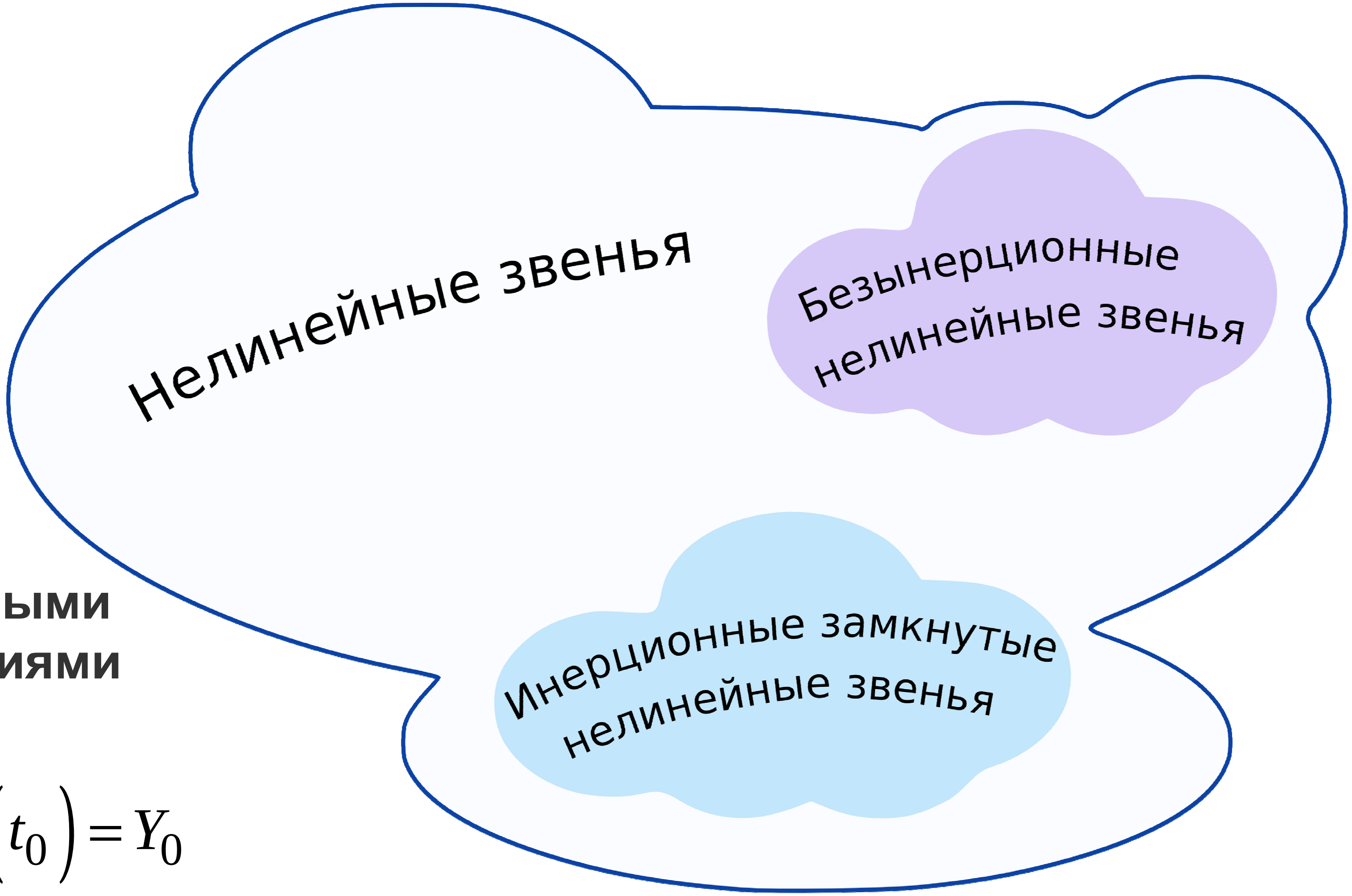


Литература

Дьяконов В. П.
MATLAB 7.*/R2006/R2007: Самоучитель. – М.: ДМК Пресс, 2008. – 768 с.: ил.

Урок 8. Программные средства численных методов

Нелинейные звенья



В общем случае описываются нелинейными дифференциальными уравнениями

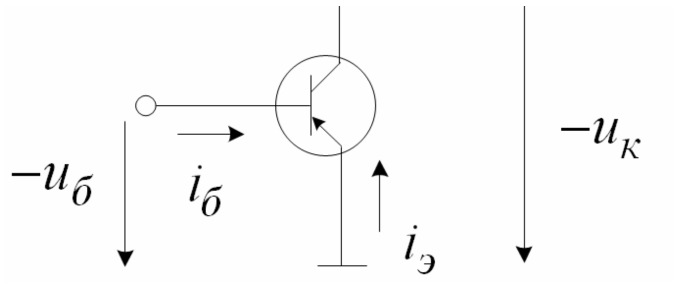
$$\frac{dY}{dt} = F(t, Y),\ Y(t_0) = Y_0$$

Но есть частные случаи, для которых задача упрощается:

- **безынерционные звенья (исчезают производные);**
- **инерционные замкнутые (работает курс Радиоавтоматики).**

Безынерционные звенья

Дано звено



Производит нелинейное преобразование входного сигнала

$$i = I_s \left(\exp\left(u/_{nV} \right) - 1 \right)$$

В общем случае вида: $y(t) = f\left(x(t)\right)$

Моделирование сводится к:

1) формирование оси времени t_k

```
T = 1e-3; tmin = 0; tmax = 100*T;
t = tmin:T:tmax;
```

2) формирование отсчетов входного сигнала $x_k = x\left(t_k\right)$

```
A = 10; f0 = 1e2;
x = A * sin( 2*pi * f0 * t );
```

3) функциональное преобразование каждого отсчета $y_k = f\left(x_k\right)$

```
nV = 0.3; Is = 10;
y = Is * (exp(y/nV) - 1);
```

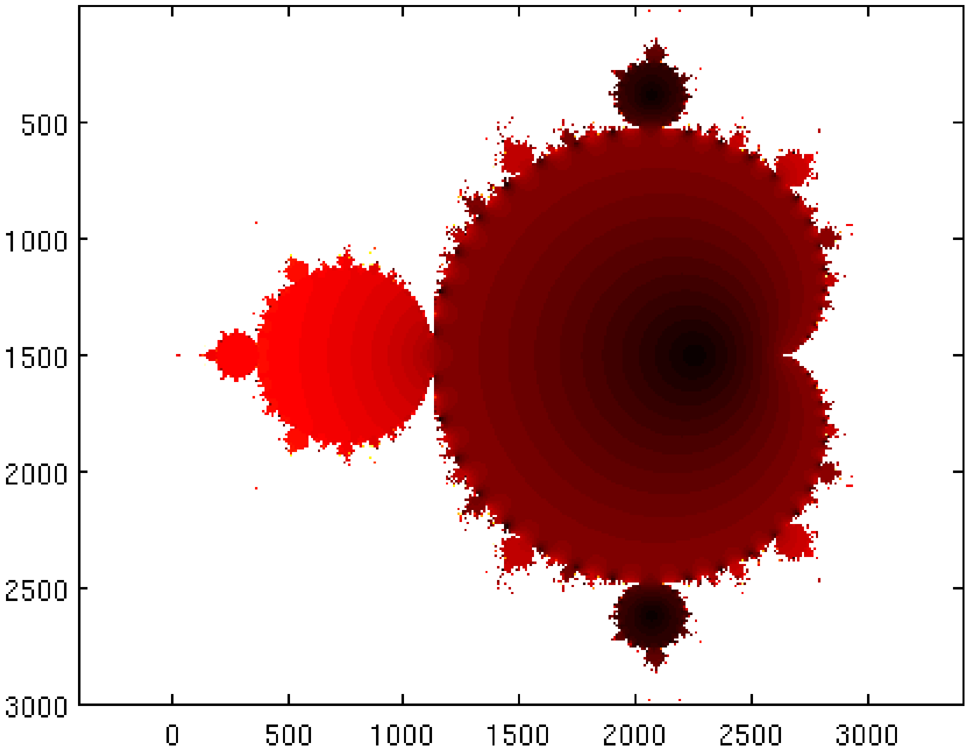
Нелинейные преобразования отсчетов производятся с помощью встроенных или библиотечных функций

Множество Мандельброта

- множество точек *c* на комплексной плоскости, для которых последовательность

$$z_{n+1} = z_n^2 + c,\ \ z_0 = 0$$

сходится




```
clear all; clc; close all
pxls=3000; N=50;

z = zeros(pxls,pxls);
c=...
repmat(linspace(-1.5,0.5,pxls),pxls,1) ...
+ 1i* repmat(linspace(-
1,1,pxls)',1,pxls);

for j=1:N
   z = z.^2 + c;
end
thresh = 5;
ind = find(real(z) > thresh);
z(ind) = thresh + 1i*imag(z(ind));
ind = find(real(z) < -thresh);
z(ind) = -thresh + 1i*imag(z(ind));
ind = find(imag(z) > thresh);
z(ind) = real(z(ind)) + thresh*1i;
ind = find(imag(z) < -thresh);
z(ind) = real(z(ind)) - thresh*1i;

colormap(hot);
imagesc(log( abs(z) + 1 )); axis equal
```

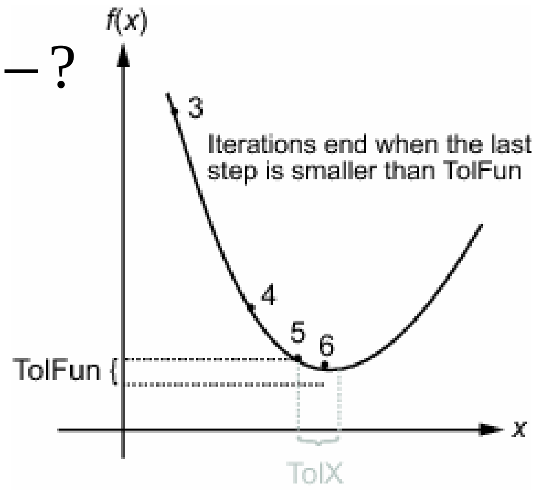
fsolve()

Часто возникает задача поиска корня системы нелинейных уравнений

$$F(X)=0,\ \ X=\{x_m\}_{m=1}^{M},\quad X-?$$

$$z=\operatorname{sinc}\left(\sqrt{x^2+y^2}\right)$$

$$z=1\leftrightarrow x,y-?$$



fsolve

Solve system of nonlinear equations

Equation

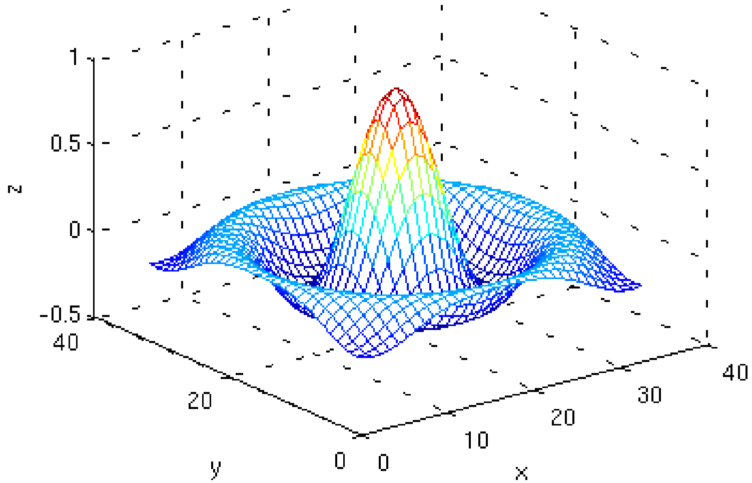
Solves a problem specified by

$F(x)=0$

for x, where x is a vector and $F(x)$ is a function that returns a vector value.

Syntax

```
x = fsolve(fun,x0)
x = fsolve(fun,x0,options)
```



main.m

```
X0 = [0.5; 1];
optnew = optimset('TolFun', 1e-20);
X = fsolve(@F, X0, optnew)
```

F.m

```
function f = F( X )
    f = sinc(sqrt(X(1)^2 + X(2)^2)) - 1;
end
```

$$X_1=x;\ \ X_2=y;$$

$$F(X)=\operatorname{sinc}\left(\sqrt{X_1{}^2+X_2{}^2}\right)-1$$

Результат:

```
X =  1.0e-08 *   -0.0095
                 -0.6332
```

fminsearch()

Схожая задача – поиск минимума/максимума функции

$$F(X),\ X=\{x_m\}_{m=1}^{M},\ \ X_0=\arg\min(F(X))-?$$

$$z=\mathrm{sinc}\left(\sqrt{x^2+y^2}\right)$$

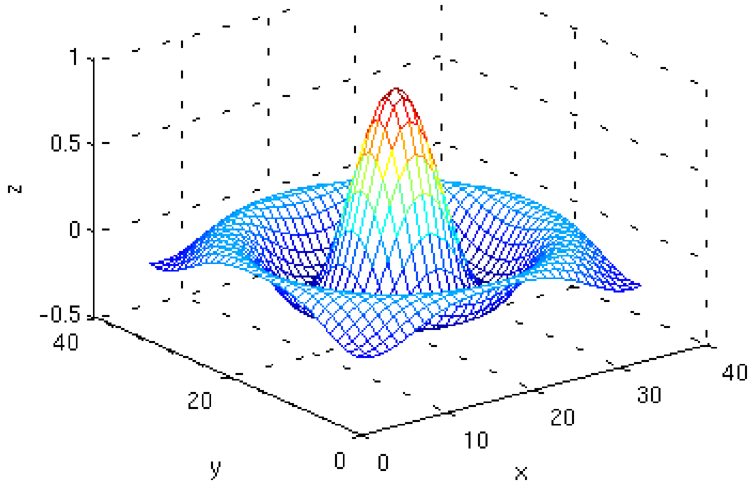
$$\max(z)\leftrightarrow x,y-?$$

fminsearch

Find minimum of unconstrained multivariable function using derivative-free method

Syntax

```
x = fminsearch(fun,x0)
x =
fminsearch(fun,x0,options)
```



main.m

```
X0 = [0.5; 1];
optnew = optimset('TolFun', 1e-20);
X = fminsearch(@F, X0, optnew)
```

F.m

```
function f = F( X )
    f = -sinc(sqrt(X(1)^2 + X(2)^2));
end
```

$$X_1=x;\ \ X_2=y;$$

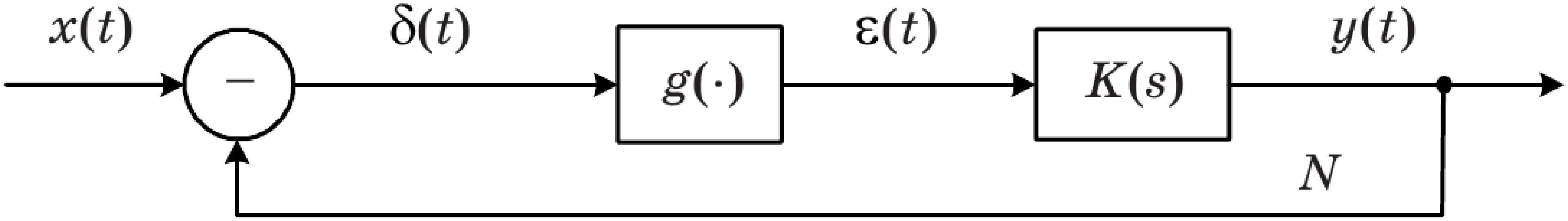
$$F(X)=-\mathrm{sinc}\left(\sqrt{X_1{}^2+X_2{}^2}\right)$$

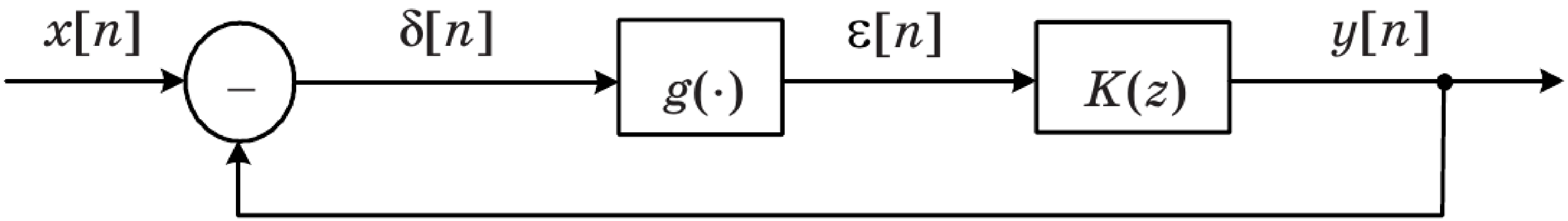
Результат:

```
X =  1.0e-07 *   -0.1338
                 -0.0701
```

Замкнутые звенья

Замкнутое инерционное нелинейное звено – система с ОС с неинерционным нелинейным звеном





$\delta_n = x_n - y_n,\ \varepsilon_n = g\left(\delta_n\right) = g\left(x_n - y_n\right),$ **- выход «дискриминатора»**

$a_N y_{n-N} + \ldots + a_0 y_n = b_M \varepsilon_{n-M} + \ldots + b_0 \varepsilon_n$ **- фильтр**

$$a_0 y_n - b_0 \varepsilon_n = b_M \varepsilon_{n-M} + \ldots + b_1 \varepsilon_{n-1} - a_N y_{n-N} - \ldots - a_1 y_{n-1}$$

$$a_0 y_n - b_0 g\left(x_n - y_n\right) = b_M \varepsilon_{n-M} + \ldots + b_1 \varepsilon_{n-1} - a_N y_{n-N} - \ldots - a_1 y_{n-1}$$

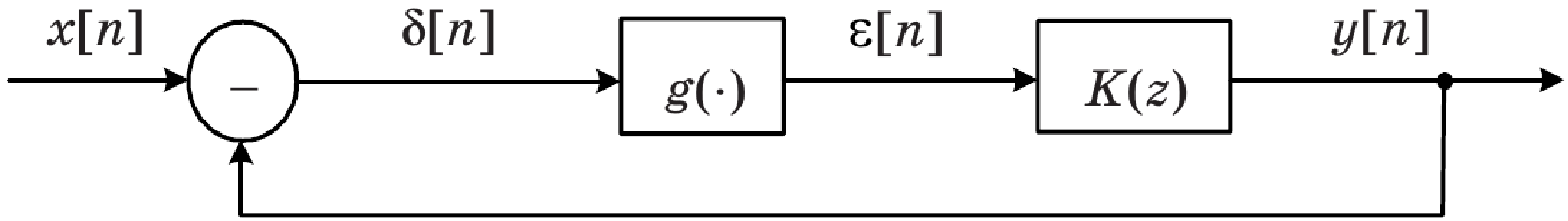
Замкнутые звенья

Получаем:

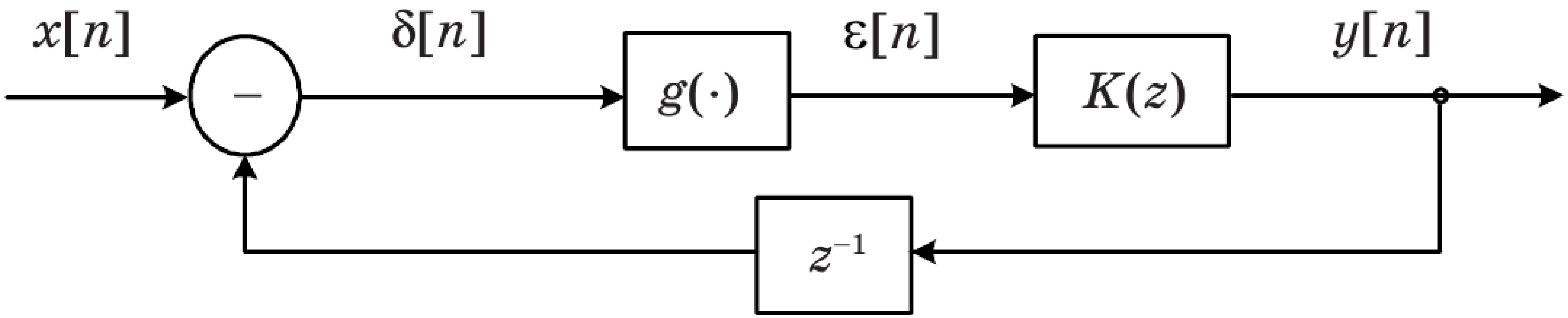
$$a_0 y_n - b_0 g\left(x_n - y_n\right) = \xi_n, \qquad \xi_n = b_M \varepsilon_{n-M} + \ldots + b_1 \varepsilon_{n-1} - a_N y_{n-N} - \ldots - a_1 y_{n-1}$$

нелинейное уравнение, возможно трансцендентное

известное к моменту n число



lifehack



$$a_0 y_n = \xi_n + b_0 g\left(x_{n-1} - y_{n-1}\right)$$

Метод Эйлера

В общем случае требуется решение системы нелинейных дифференциальных уравнений

$$\frac{dY}{dt} = F(t,Y),\ Y(t_0) = Y_0,\ Y(t) = \left\{ y_m(t) \right\}_{m=1}^{M}$$

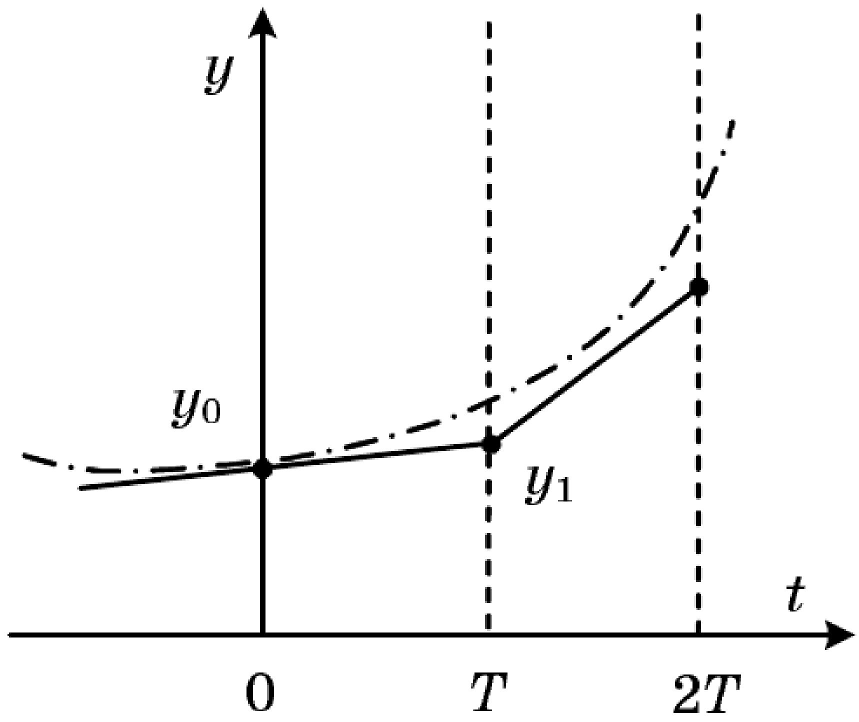
Есть множество методов решения дифуров. Особую роль играют методы конечных разностей, основанные на замене производных разностными схемами.

Метод Эйлера

Идея – заменить производную по времени на конечную разность

$$Y_n = Y_{n-1} + T \cdot F\left(t_{n-1}, Y_{n-1}\right)$$

```
y = y0;
for t = tmin:T:tmax;
   y = y + T*F(t,y);
   ...
end
```



Ошибка пропорциональна T^2

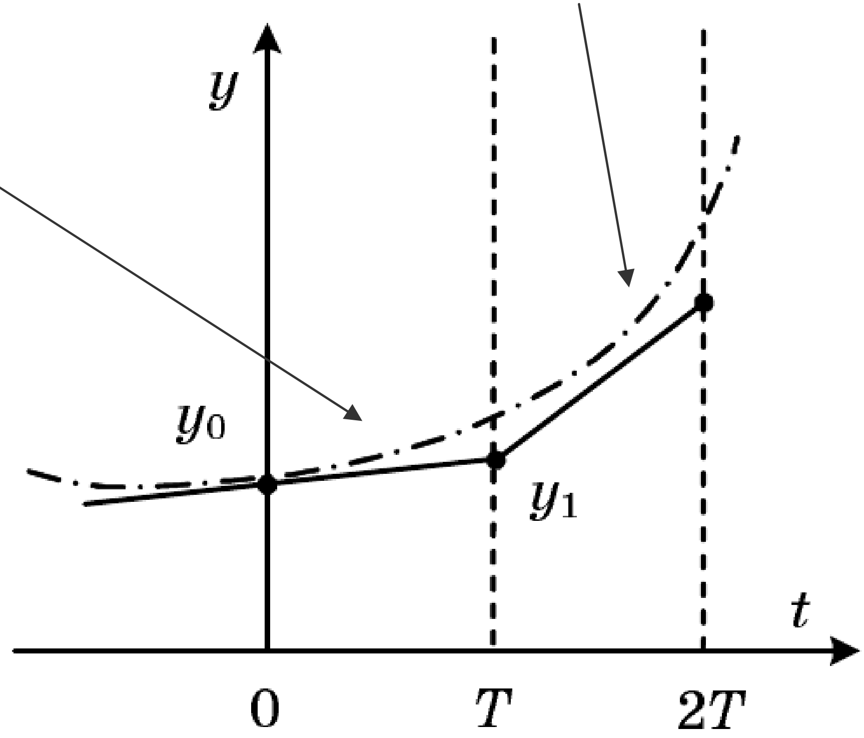
Расширенный Эйлера

Расширенный метод Эйлера

$$Y_n = Y_{n-1} + \frac{T}{2} \cdot \left\{ F\left(t_{n-1}, Y_{n-1}\right) + F\left(t_n, \tilde{Y}_n\right) \right\} \qquad \tilde{Y}_n = Y_{n-1} + T \cdot F\left(t_{n-1}, Y_{n-1}\right)$$

Идея – выбрать наклон как среднее между наклоном на прошлом шаге и наклоном по простому методу Эйлера

```
y = y0;
for t = tmin:T:tmax;
   yt = y + T*F(t,y);
   y = y + T/2*(F(t,y) + F(t, yt));
   ...
end
```



Ошибка пропорциональна T^3

Рунге-Кутта

Метод Рунге-Кутта 4 порядка

$$Y_{n+1} = Y_n + \frac{T}{6}\cdot\left\{K_0 + 2K_1 + 2K_2 + K_3\right\}$$

$$K_0 = F\left(nT, Y_n\right)$$

$$K_1 = F\left(\left(n+\frac{1}{2}\right)T,\ Y_n + \frac{1}{2}K_0\right)$$

$$K_2 = F\left(\left(n+\frac{1}{2}\right)T,\ Y_n + \frac{1}{2}K_1\right)$$

$$K_3 = F\left((n+1)T,\ Y_n + K_2\right)$$

Ошибка пропорциональна T^5

Пример:

$$y'' + 4y = \cos 3t,\ y(0) = 0.8,\ y'(0) = 2$$

После замены переменных:

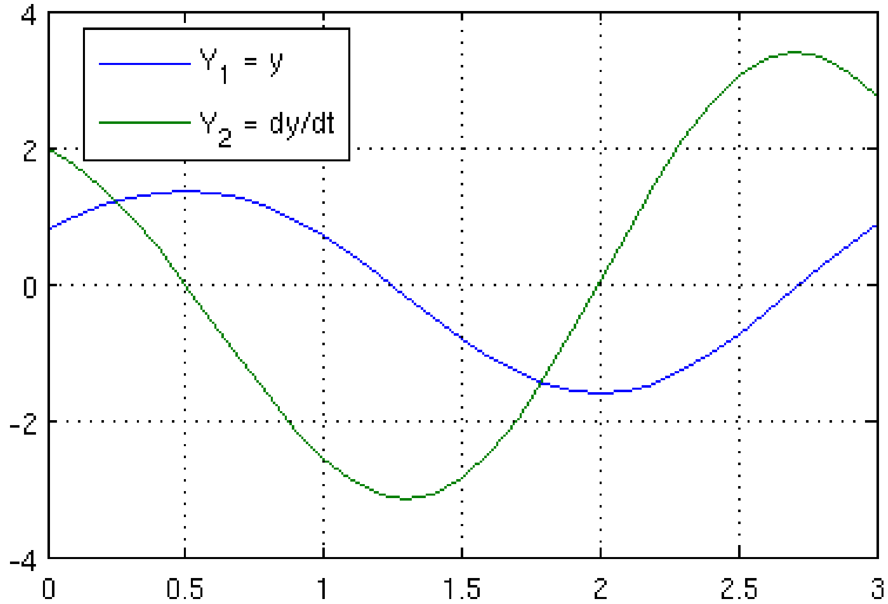
$$\begin{cases} y_1' = y_2, \\ y_2' = \cos 3t - 4y_1 \end{cases}$$

main.m:

```
T = 0.1; tmin = 0; tmax = 3;
Y0 = [0.8 2]; % Начальные условия
[t, Y] = ode45('diffs', tmin:T:tmax, Y0);
plot(t, Y); grid on
legend('Y_1 = y', 'Y_2 = dy/dt');
```

diffs.m:

```
function dY = diffs( t, Y )
dY = Y(:);
dY(1) = Y(2);
dY(2) = cos(3*t) - 4*Y(1);
end
```



Решатели

- **ode45** – одношаговые явные методы Рунге Кутта 4 го и 5 го порядков. Использовать в первую очередь. Дает хорошие результаты, если система нежесткая.
- **ode23** – одношаговые явные методы Рунге Кутта 2 го и 4 го порядков. Может дать выигрыш
в скорости решения.
- **ode113** – многошаговый метод Адамса–Башворта–Мултона переменного порядка класса предиктор–корректор. Это адаптивный метод, который может обеспечить высокую точность решения.
- **ode15s** – многошаговый метод переменного порядка. Применять, если ode45 не справилась.
- **ode23s** – одношаговый метод, использующий модифицированную формулу Розенброка 2 го порядка. Высокая скорость, низкая точность, жесткая система
- **ode23t** – неявный метод трапеций с интерполяцией. Для колебательных систем.
- **ode23tb** – неявный метод Рунге–Кутта, альтернатива ode15s.
- **bvp4c** – для уравнений вида y’ = f(t,y), F(y(a), y(b),p) = 0 (полная форма системы уравнений Коши). Для задания граничных условий как в начале, так и в конце интервала решения.
- **pdepe** – служит для решения систем параболических и эллиптических дифференциальных уравнений в частных производных.

Аттрактор Лоренца

Описывается системой уравнений:

$$y_1' = -\sigma y_1 + \sigma y_2,$$

$$y_2' = r y_1 - y_2 - y_1 y_3,$$

$$y_3' = y_1 y_2 - b y_3$$

$$r = 28;\ \sigma = 10;\ b = 8/3;$$

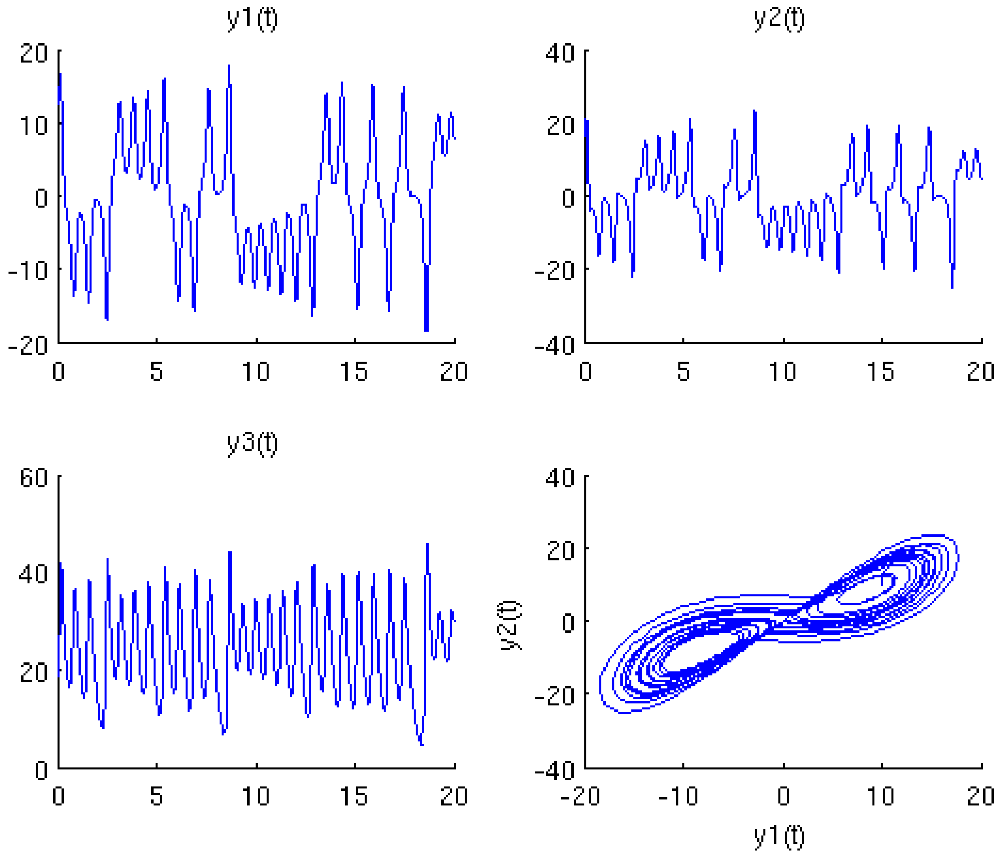
$$y_1(0) = 10,\ y_2(0) = 10,$$

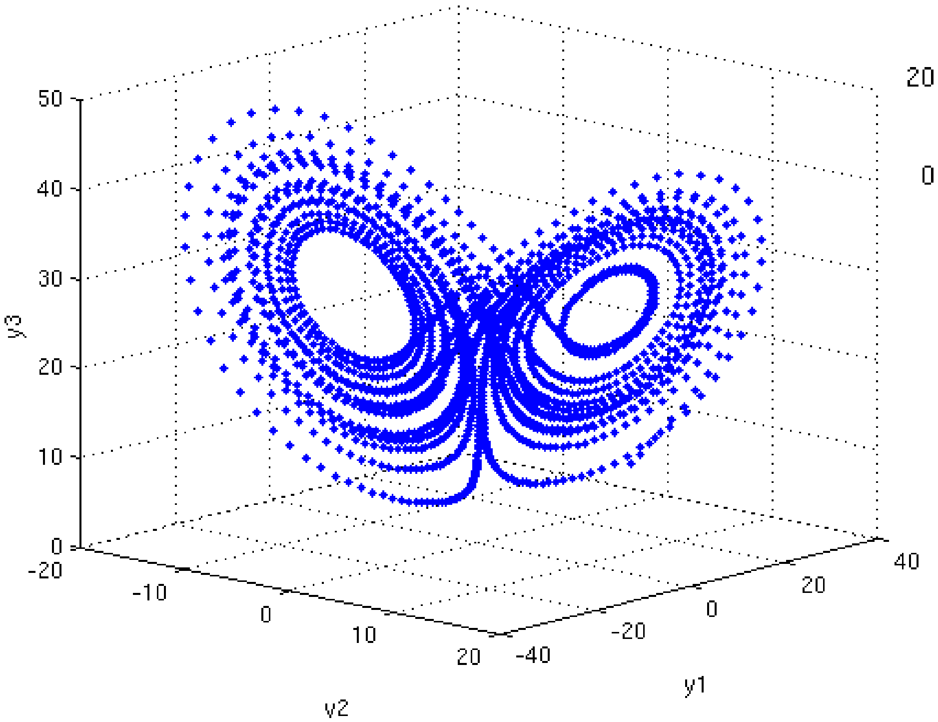
$$y_3(0) = 10,$$

```
function F = loren(t, y)
r = 28;
sigma = 10;
b = 8/3;
F = [sigma*(y(2)-y(1));
     r*y(1)-y(2)-y(1)*y(3);
     y(1)*y(2)-b*y(3)];
```

```
clear all; clc; close all
% Начальные условия
Y0 = [10;10;10];

T = 0.01;
tmin = 0; tmax = 20;

% Решение методом Рунге-Кутта 4-5 пор.
[t, Y] = ode45(@loren, tmin:T:tmax, Y0);

figure(1)
subplot(2,2,1); plot(t, Y(:,1)); title('y_1(t)');
subplot(2,2,2); plot(t, Y(:,2)); title('y_2(t)');
subplot(2,2,3); plot(t, Y(:,3)); title('y_3(t)');
subplot(2,2,4); plot(Y(:,1),Y(:,2));
xlabel('y_1(t)'); ylabel('y2_(t)');

figure(2)
plot3(Y(:,1), Y(:,2), Y(:,3), '.');
grid on; xlabel('y_2')
ylabel('y_1'); zlabel('y_3');
```

Аттрактор Лоренца





Аттрактор Лоренца

**Измените на 1%
начальные условия:**

```
Y0 = [10;10;10] * (1+1/100);
```