



**AGH**

AKADEMIA GÓRNICZO-HUTNICZA  
IM. STANISŁAWA STASZICA W KRAKOWIE

# Inżynierskie techniki obliczeniowe

## 2021 / 2022

Wykład nr 6

**Dr inż. Przemysław Korohoda**  
**E-mail:** korohoda@agh.edu.pl  
**Tel.wewn.AGH:** (012-617)-27-52  
**Pawilon C3 - p.506**

**Strona WWW:**

[home.agh.edu.pl/~korohoda/rok\\_2021\\_2022\\_lato/ITO\\_EL\\_1](http://home.agh.edu.pl/~korohoda/rok_2021_2022_lato/ITO_EL_1)

**UPeL:** ITOEL2022

# Plan wykładu

## Różne przykłady

# Przykład zadania

Proszę opracować „detektor liczb pierwszych”.  
Ważne elementy: 1) własna funkcja, 2) grafika, 3) sprawdzenie poprawności.

```
% m-plik skryptowy: test_czy_pierwsza.m
%
% Autor i kiedy opracował;
% Kto i kiedy uzupełnił/poprawił;
%
clc; clear; close all;

N=100; x=2:N;

for k=1:N-1,
    t(k)=czy_liczba_pierwsza(x(k));
end;

x=x(t==1);

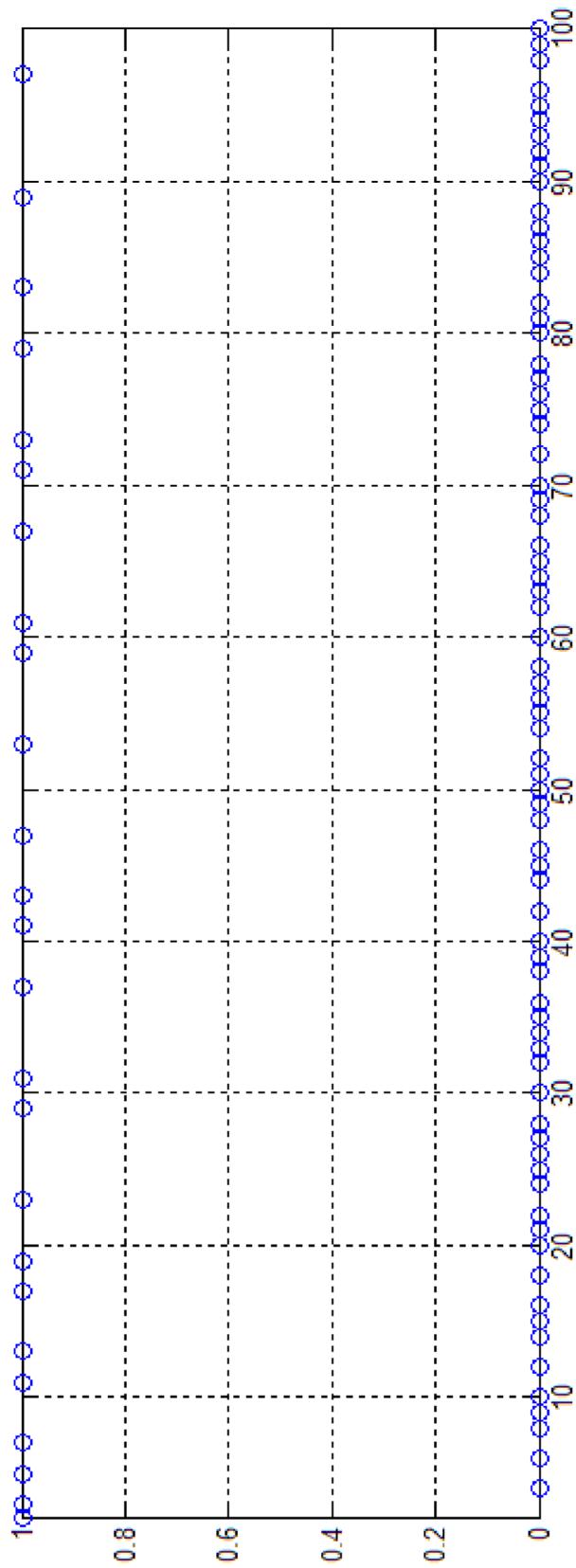
figure(1); clf;
plot(2:N,t,'bo'); grid on; axis tight;

%
% KONIEC PLIKU;
```

# Przykład zadania (cd.)

**Wyniki:**

$x =$	2	3	5	7	11	13	17	19	23	29	31	37	41	43	47	53	59
61	67	71	73	79	83	89	97										



# Inne warianty rozwiązań

```
function t=czy_liczba_pierwsza_2(x,wersja);
% t=czy_liczba_pierwsza_2(x,wersja);
%
% Autor i kiedy opracował;
% Kto i kiedy uzupełnił/poprawił;

switch wersja,
case 1,
t=1;
for n=2:x-1,
if x/n==round(x/n); t=0; end;
end;
case 2,
t=1;
for n=2:sqrt(x),
if x/n==round(x/n); t=0; end;
end;
case 3,
if min(rem(x,2:sqrt(x)))==0; t=0; else t=1; end;
case 4,
if min(mod(x,2:sqrt(x)))==0; t=0; else t=1; end;
end;

%
% KONIEC FUNKCJI;
```

## Inne warianty rozwiązań - weryfikacja

```
% m-plik skryptowy: test_funkcji_czy_liczba_pierwsza.m
%
% Autor i kiedy opracował;
clc; clear; close all;

N=100; x=2:N;

for k=1:N-1,
t1(k)= czy_liczba_pierwsza_2(x(k),1);
t2(k)= czy_liczba_pierwsza_2(x(k),2);
t3(k)= czy_liczba_pierwsza_2(x(k),3);
t4(k)= czy_liczba_pierwsza_2(x(k),4);
end;

x1=x(t1==1); x2=x(t2==1); x3=x(t3==1); x4=x(t4==1);

test_porownawczy_12 = max(abs(x1(:)-x2(:)));
test_porownawczy_13 = max(abs(x1(:)-x3(:)));
test_porownawczy_14 = max(abs(x1(:)-x4(:))), x1,

figure(1); clf;
plot(2:N,t1,'bo'); grid on; axis tight;

% KONIEC PLIKU;
```

# Wzór Taylora

```

.....
clc; clear; close all;

N=15;
x=-5:0.001:5; Nx=length(x);

x0=2; a=2*pi*0.4;

y_ref=cos(a*x); y0=cos(a*x0);

y=ones(1,Nx)*y0;
for n=1:N,
    y=y + dfx(x0,a,n)/factorial(n) * (x-x0).^n;
end;

figure(1);
plot(x,y_ref,'b.-'); grid on; hold on;
plot(x,y,'r.');
plot(x0,y0,'ko');
ylim([-2,2]);

% KONIEC PLIKU;

```

```

function y=dfx(x,a,n)

if n/2==round(n/2),
y=(-1)^(n/2)*cos(a*x)*a^n;
else
y=(-1)^((n+1)/2)*sin(a*x)*a^n;
end;

% KONIEC FUNKCJI

```

$$y(x) = \cos(2 \cdot \pi \cdot f_0 \cdot x)$$

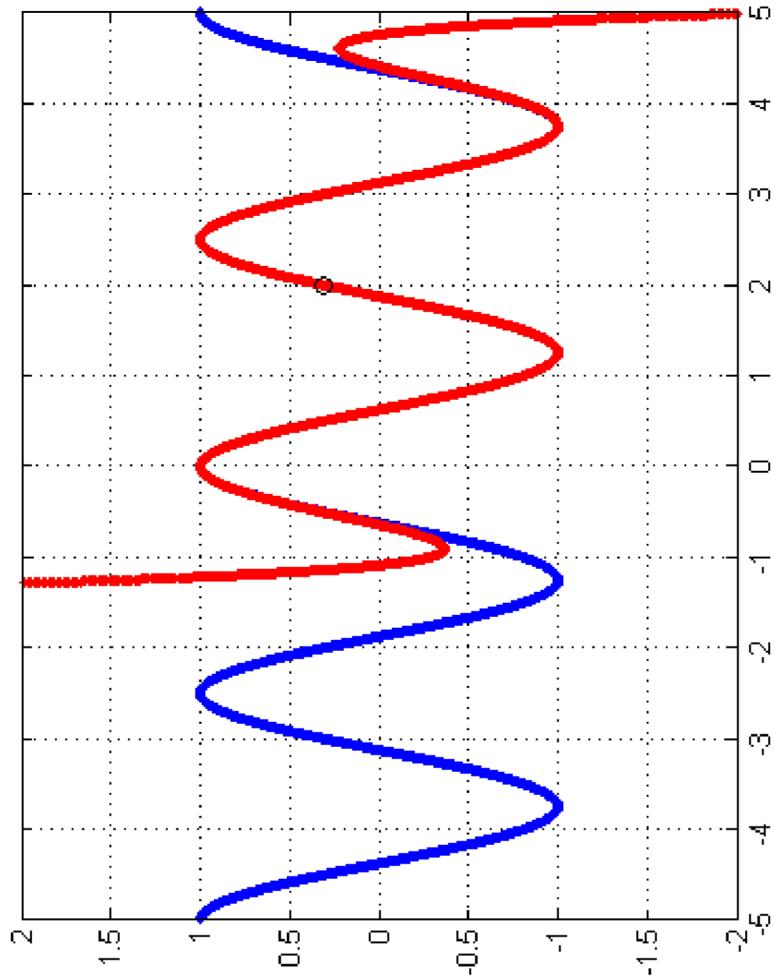
$$y(x) \approx \sum_{n=0}^N \frac{y^{(n)}(x_0)}{n!} \cdot (x - x_0)^n$$

# Wzór Taylora - wykres

```
figure(1);
plot(x,y_ref,'b.-'); grid on; hold on;
plot(x,y_r,'r.');
plot(x0,y0,'ko');
ylim([-2,2]);
```

$$y(x) \approx \sum_{n=0}^N \frac{y^{(n)}(x_0)}{n!} \cdot (x - x_0)^n$$

$$y(x) = \cos(2 \cdot \pi \cdot f_0 \cdot x)$$



# Szereg Fouriera

```

.....
clc; clear; close all;
T=6; dt=T/1000;
t=-T/2:dt:T/2; Nt=length(t); fT=1/T;
Nmax=20;
Ns=4;
ts=rand(1,Ns)*T-T/2;
xs=rand(1,Ns);

x=rand*ones(1,Nt);
for ns=1:length(ts);
    x(t>ts(ns))=xs(ns);
end;

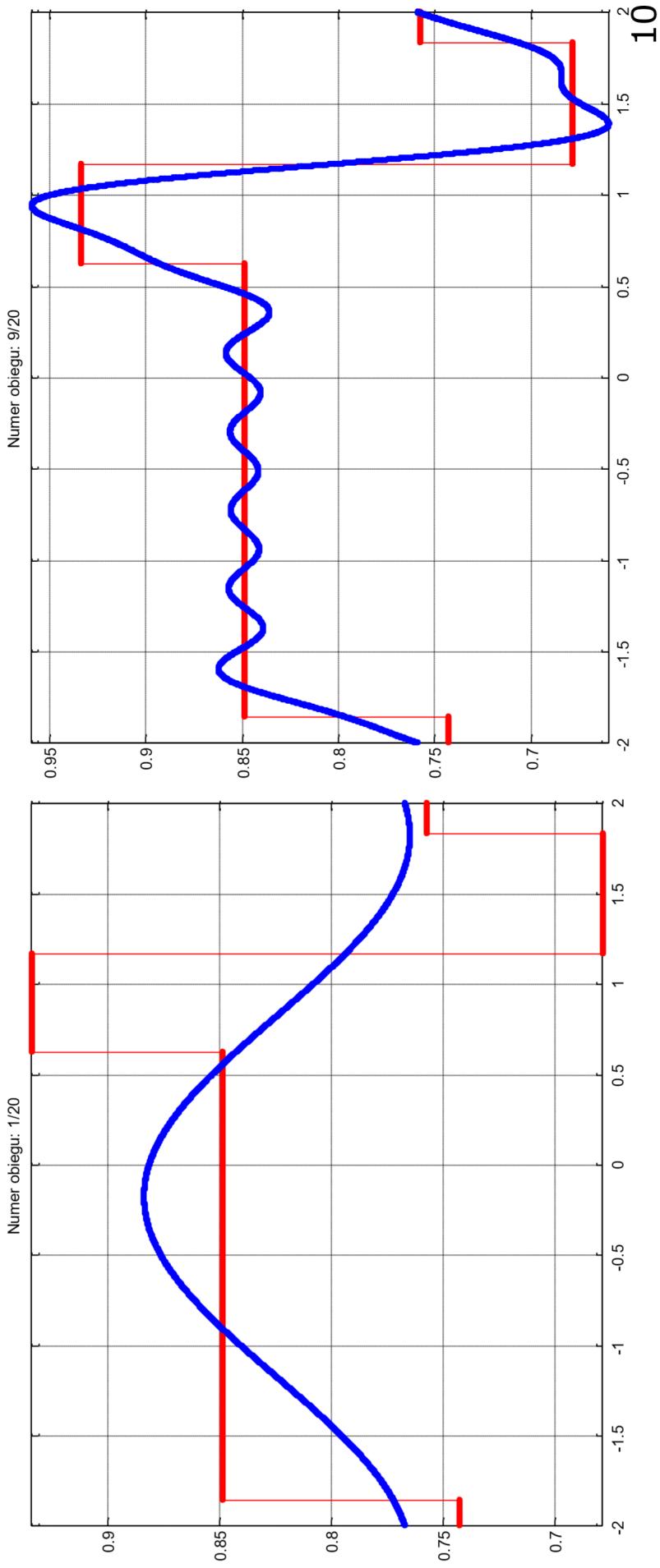
for N=1:Nmax;
    x1=zeros(1,Nt);
    for n=-N:N,
        xbn = exp(j*2*pi*fT*n*t);
        y = x.*conj(xbn);
        cn = (1/T)*(sum(y(2:end-1))+(y(1)+y(end))/2)*dt;
        x1 = x1+cn*xbn;
    end;
    % tu należy wstawić tworzenie wykresu;
end;
% KONIEC PLIKU;

```

9

# Szereg Fouriera - tworzenie wykresu

```
figure(1); clf;
plot(t,x,'r.-'); grid on; hold on;
plot(t,x1,'b.-');
title(['Numer obiegu: ',num2str(N),' ',num2str(Nmax)]); axis tight;
pause;
```



# Objętość bryły obrotowej

```
....  
clc; clear; close all;
```

% przepis na funkcje:

```
funkcja=1;  
switch funkcja,  
case 1, f=inline('0.3*x','x'); %/ sciezty stożek (dla dodatnich xa oraz xb, xb>xa);  
case 2, f=inline('sqrt(1^2-(x-2).^2)','x'); %/ powinna byc kula;  
....  
end;
```

```
xa=1; xb=3; %/ granice wzdluz osi x (osi obrotu);  
dx=(xb-xa)/1000;  
x=xa:dx:xb;
```

```
V=sum(pi*f(x).^2)*dx, %/ metoda prostokatow, bo klarowniejszy wzor;
```

```
if funkcja==1, V_ref=(1/3)*pi*(xb*f(xb)^2-xa*f(xa)^2), end;  
%/ dla dodatnich xa oraz xb;  
if funkcja==2, V_ref=(4/3)*pi*((xb-xa)/2)^3, end;
```

%/ tu ma być część graficzna;

%/ KONIEC PLIKU;

# Objętość bryły obrótowej (cd.)

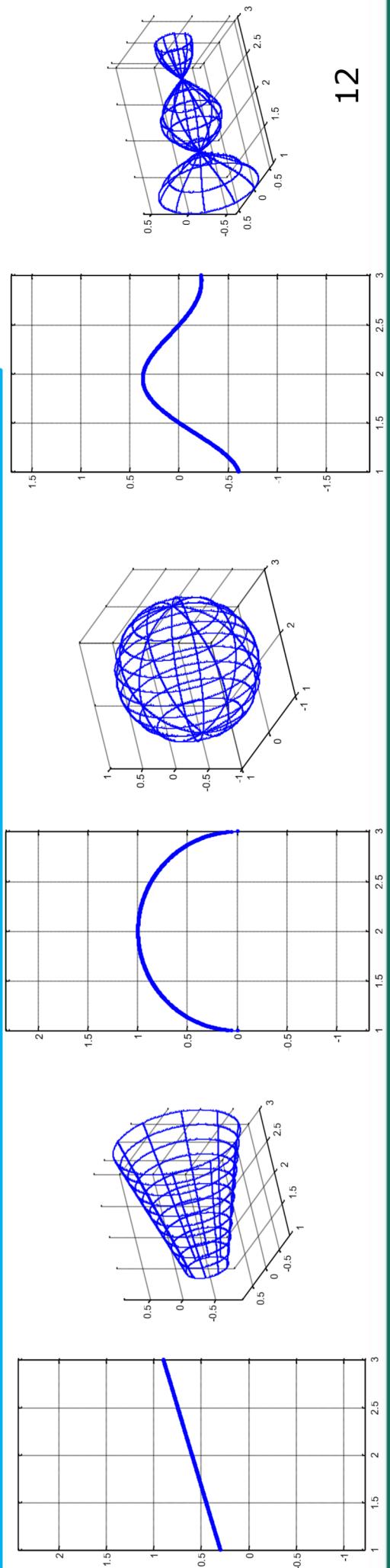
% to jest ta część graficzna;

```

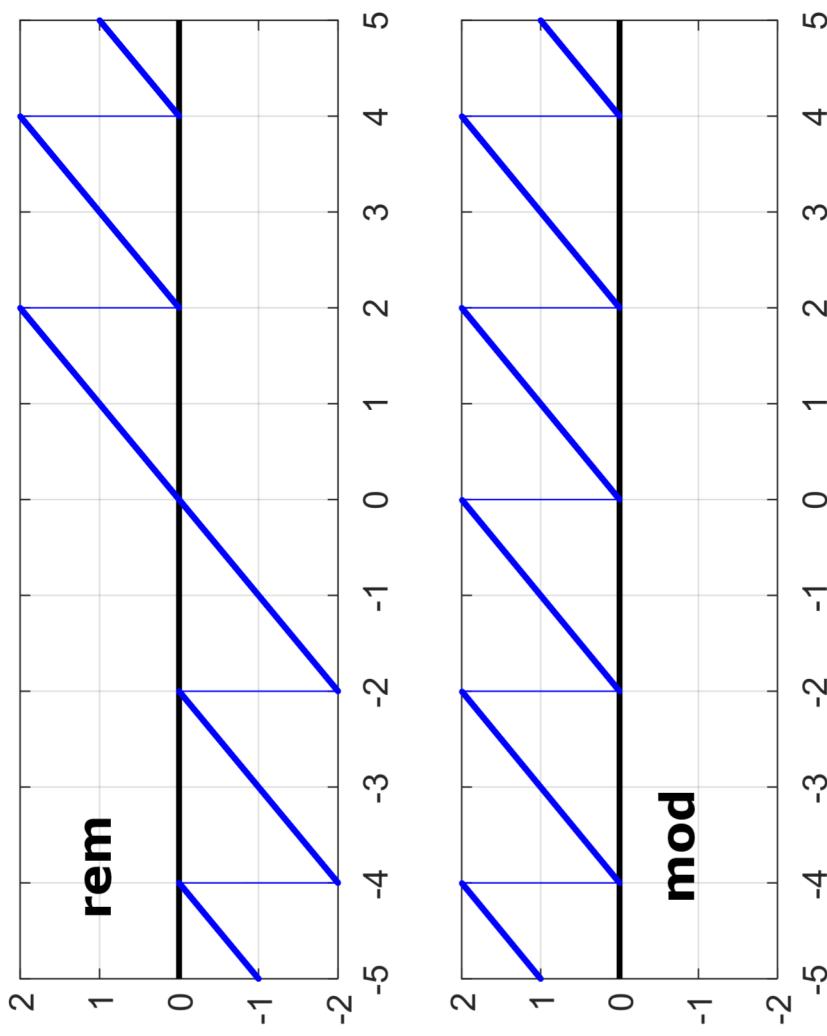
figure(1);
subplot(1,2,1);
plot(x,f(x),'b.-'); grid on; axis equal;

subplot(1,2,2);
dphi=(2*pi)/10; dxx=(xb-xa)/10;
Np=length(phi);
hold on;
for phi=0:dphi:2*pi;
    plot3(x , f(x)*sin(phi) , f(x)*cos(phi) , 'b-');
end;
for xx=xa:dxx:xb;
    plot3(ones(1,Np)*xx , f(xx)*sin(phi) , f(xx)*cos(phi) , 'b-');
end;
grid on; view([-30,25]);
axis equal;

```

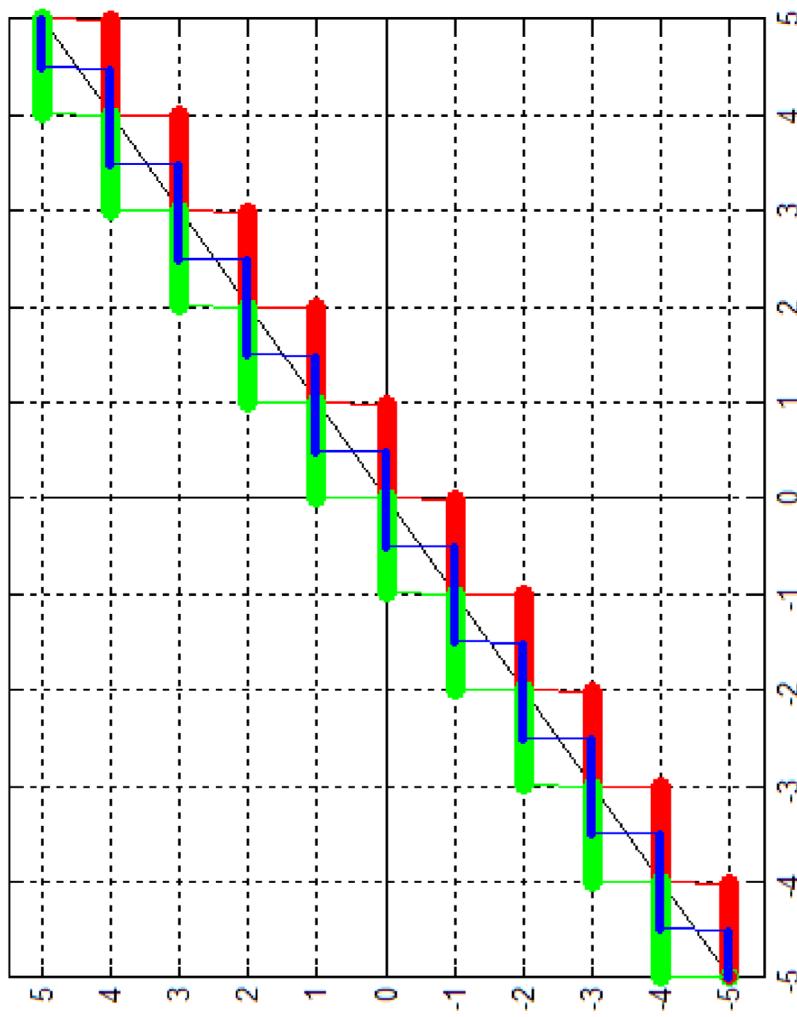


# Modulo/reszta, zaokrąglenia, kwantyzacja



```
dx=0.01;  
x=-5:dx:5;  
y1=rem(x,2);  
y2=mod(x,2);  
y3=floor(x);  
y4=ceil(x);  
y5=round(x);  
y6=floor(x)+0.5;
```

# Modulo/reszta, zaokrąglenia, kwantyzacja



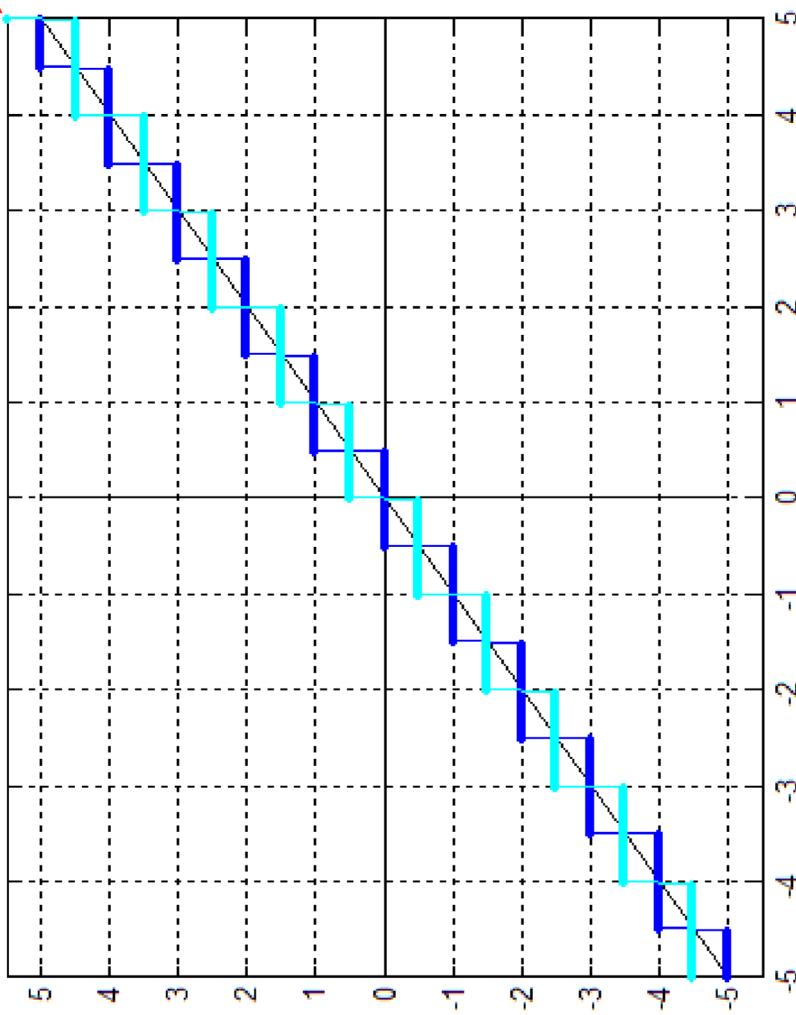
```
.....
dx=0.01;
x=-5:dx:5;
y1=rem(x,2);
y2=mod(x,2);
y3=floor(x);
y4=ceil(x);
y5=round(x);
y6=floor(x)+0.5;
....
```

**Rysowanie linii pomocniczych:**

```
.....
plot([x(1),x(end)], [x(1),x(end)],'k','LineWidth',lw1); grid on; hold on;
plot([x(1),x(end)], [0,0], 'k','LineWidth',lw1);
plot([0,0],[x(1),x(end)], 'k','LineWidth',lw1);
.....
```

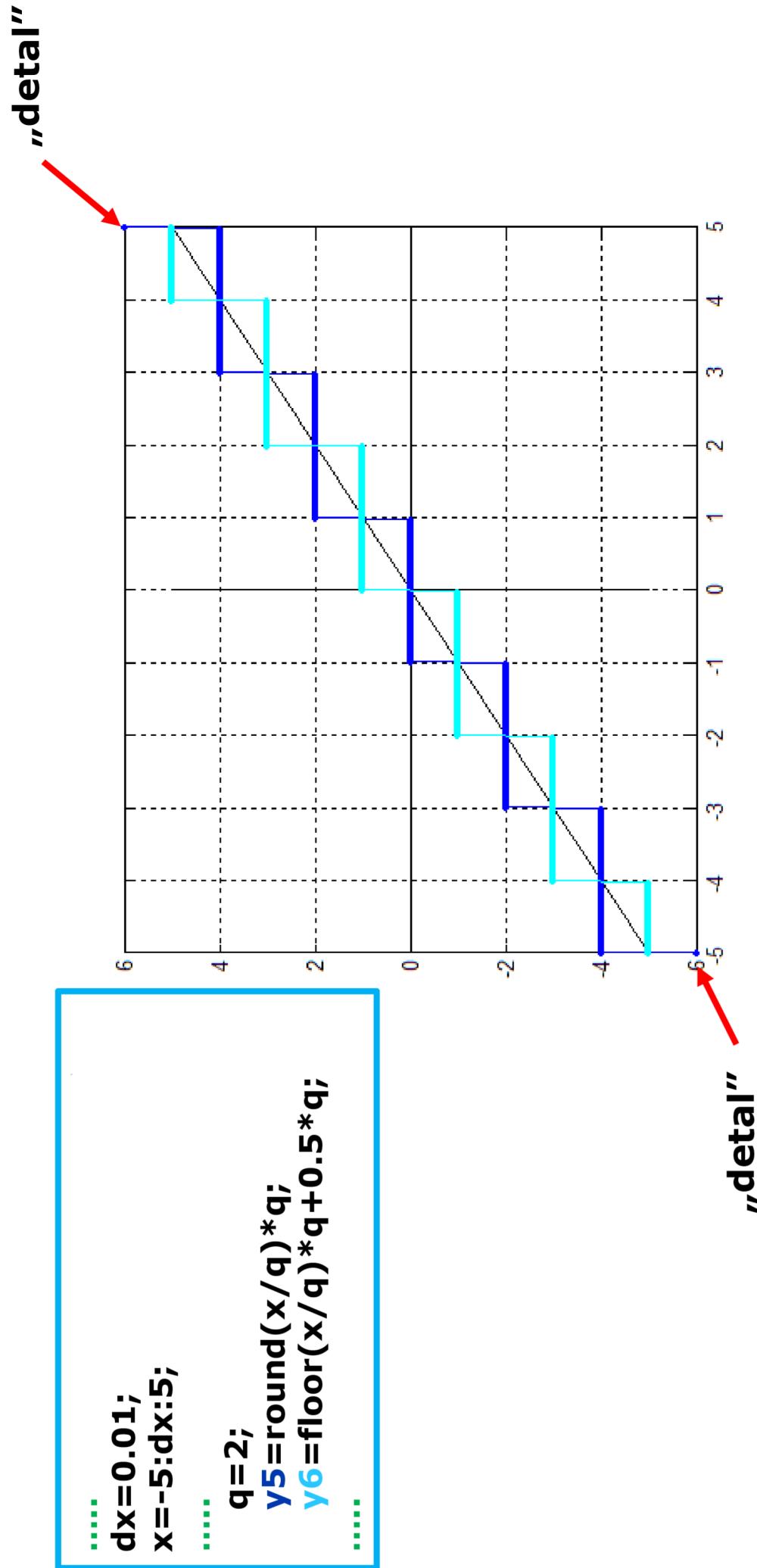
# Modulo/reszta, zaokrąglenia, kwantyzacja

„detal”



```
dx=0.01;  
x=-5:dx:5;  
y1=rem(x,2);  
y2=mod(x,2);  
y3=floor(x);  
y4=ceil(x);  
y5=round(x);  
y6=floor(x)+0.5;
```

# Modulo/reszta, zaokrąglenia, kwantyzacja



# Kody dziesiętne, binarne, heksa-

```
function y=my_dec2bin(x,B);
% y=my_dec2bin(x,B);
%
% x - calkowita liczba dodatnia;
% y - wektor 0/1;
% B - liczba bitow (gdy liczba x jest zbyt duza, to B jest ignorowane);
% (podawanie B mozna ominac);
%
% Opracowanie: P.Korohoda, 10/12/2014;

if x==0,
    y=0;
else
    N2=ceil(log2(x+1));
    y=zeros(1,N2);
    for n=N2-1:-1:0,
        if x>=2^n, y(N2-n)=1; x=x-2^n; end;
        if x==0, break; end;
    end;
end;

if nargin==2,
    By=length(y);
    if B>By,
        y=[zeros(1,B-By),y];
    end;
end;
%
% KONIEC FUNKCJI;
```

## Kody dziesiętne, binarne, heksa-

```
.....
S='0123456789ABCDEF'; % Symbole kodu;
% to samo, co: S=['0','1','2','3',...
Hx=[S(3),S(1),S(16),S(2),S(2)],  

N=length(Hx);  

x=0;  

for n=1:N,  

    k = find(S==Hx(n));  

    x = x+(k-1)*16^(N-n);  

end;  

x,  

test = dec2hex(x),  

....
```

## Kod uzupełnieniowy do dwóch (U2)

Zamienia liczbę dziesiętną (całkowitą) z przedziału:

$$x \in [-2^{n-1}, 2^{n-1} - 1]$$

na ciąg  $n$  bitów:

$$x \in [a_{n-1}, a_{n-2}, \dots, a_1, a_0]$$

tak, że zachodzi zależność:

$$x = -a_{n-1} \cdot 2^{n-1} + \sum_{k=0}^{n-2} a_k \cdot 2^k$$

Warto się też zastanowić nad binarną reprezentacją liczb ułamkowych.

# Kod Graya

Zamienia liczbę binarną w kodzie naturalnym na kod (również binarny), tak że kolejne liczby różnią się tylko jednym bitem:

000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100

## Kodowanie:

## Dekodowanie:

Recepta: należy przesunąć liczbę o jeden bit w prawo i wykonać operację XOR (różnicę symetryczną) na dwóch ciągach binarnych:  
a) liczby pierwotnej i b) liczby przesuniętej o bit.

Recepta: należy przyjąć jako pierwszy bit (najbardziej znaczący) kodu naturalnego pierwszy bit kodu Graya, a każdy kolejny bit wyznaczyć jako wynik operacji XOR na odpowiednim biecie kodu Graya i poprzednio wyznaczonego bitu kodu naturalnego.

# Bramki logiczne

```
function y=bramka_logiczna(x,TP);
% y=bramka_logiczna(x,TP);
%
% y - wynik (1/0);
% x - wektor wejściowy (1/0);
% TP - tabela prawdy
% (musi mieć rozmiar pasujący do wektora x
% - tj. o jedna kolumnę więcej, gdzie są wyniki);
%
% Opracowanie: P.Korohoda, 10/12/2014;
%
N=length(x);
for n=1:N,
    k=find(x(n)==TP(:,n));
    TP=TP(k,:);
end;
y=TP(1,N+1);
```

Jak ta funkcja działa?  
A może można to samo zrealizować inną? Prościej?

```
function y=xor_2(x);
% y=xor_2(x);
%
% P.Korohoda, 10/12/2014;
%
TP=[1,1,0;
    1,0,1;
    0,1,1;
    0,0,0];
y=bramka_logiczna(x,TP);
```

# Układ logiczny

Tabela prawdy całego układu:

<del><math>x_1</math></del>	<del><math>x_2</math></del>	<del><math>x_3</math></del>	<del><math>x_4</math></del>	<del><math>y</math></del>
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

```

Bity=4;
N=2^Bity-1;

for n=0:N,
    x=my_dec2bin(n,Bity);
    A(n+1,:)=x;

    y11=nand_2(x([1,2]));
    y12=nand_2(x([2,3]));
    y21=and_2([y11,y12]);
    y31=xor_2([y21,x(4)]);

    y=y31;

% to samo zapisane nieco krocej:
y1=xor_2( [and_2( [nand_2(x([1,2])),... 
            nand_2(x([2,3]))] ) ,x(4)]);

Y(n+1)=y;
Y1(n+1)=y1;
end;
test=max(abs(Y(:)-Y1(:)));
TP=[A,Y(:)],
```

# Modelowanie diody p-n

$$I_D = I_S \cdot \left( e^{U_D / \phi_T} - 1 \right)$$

```
clc; clear; close all;
q=1.6e-19; % [C];
T=300; % [K];
k=1.38e-23; % stała Boltzmanna [J/K];
UT=k*T/q;
Is=10e-14; % [A];
```

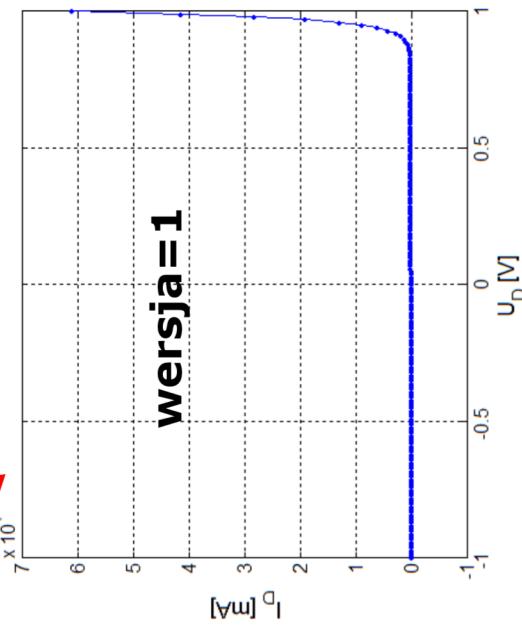
```
wersja=1;
switch wersja,
case 1, UD=-1:0.01:1;
case 2, UD=0:0.001:0.8;
case 3, UD=-1:0.01:0;
end;
```

```
ID=Is*(exp(UD/UT)-1);

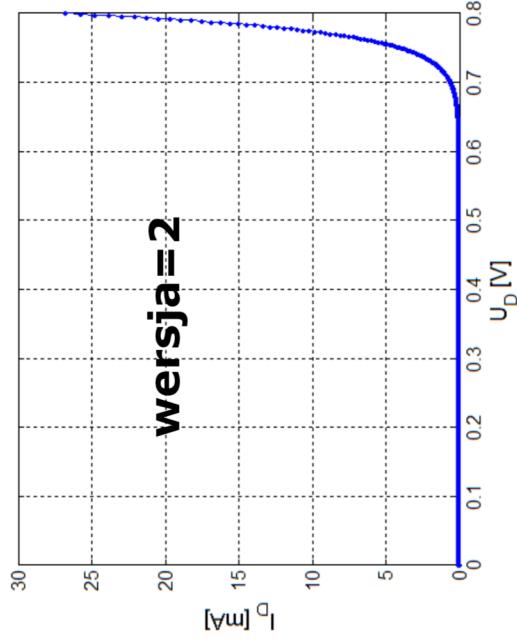
figure(2); fs=14;
plot(UD, ID*1000, 'b.-'); grid on;
xlabel('U_D [V]', 'fontsize', fs);
ylabel('I_D [mA]', 'fontsize', fs);
set(gca, 'fontsize', 12);
```

% KONIEC;

**mnożnik!**



**wersja=1**

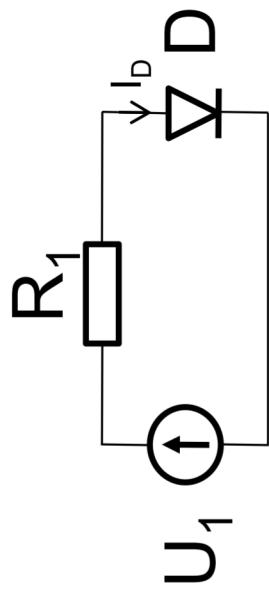


**wersja=2**



# Praca diody w układzie opornik – dioda AGH (R-D)

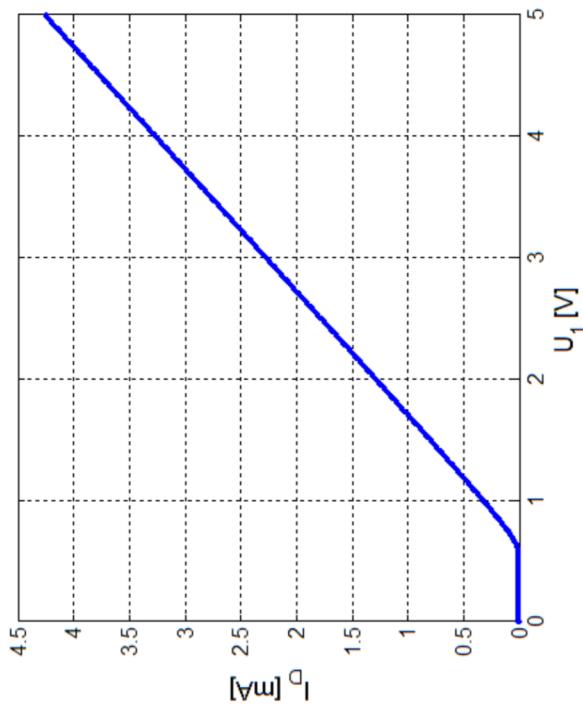
Iteracyjne wyznaczanie rozwiązania – metoda bisekcji



```
R1=1000; % [Ohm];
U1=0:0.01:5;
for k=1:length(U1),
    Ua=0; Ub=U1(k); Ux=(Ua+Ub)/2; Ierr=100;

    while abs(Ierr)>1e-10,
        Ix=Is*(exp(Ux/UT)-1);
        Ierr=Ix-(U1(k)-Ux)/R1;
        if Ierr>0
            Ub=Ux;
        else
            Ua=Ux;
        end;
        Ux=(Ua+Ub)/2;
    end;

    ID(k)=Ix;
end;
```



Uwaga – powyższa realizacja bisekcji jest poprawna tylko dla dodatnich U1

# Odpowiedź prądu diody na wymuszenie kosiinusoidalne w układzie R-D

## Iteracyjne wyznaczanie rozwiązania – metoda bisekcji (cd.)

```

f1=100; tmax=3/f1; dt=tmax/1000; t=0:dt:tmax;
U1=cos(2*pi*f1*t);

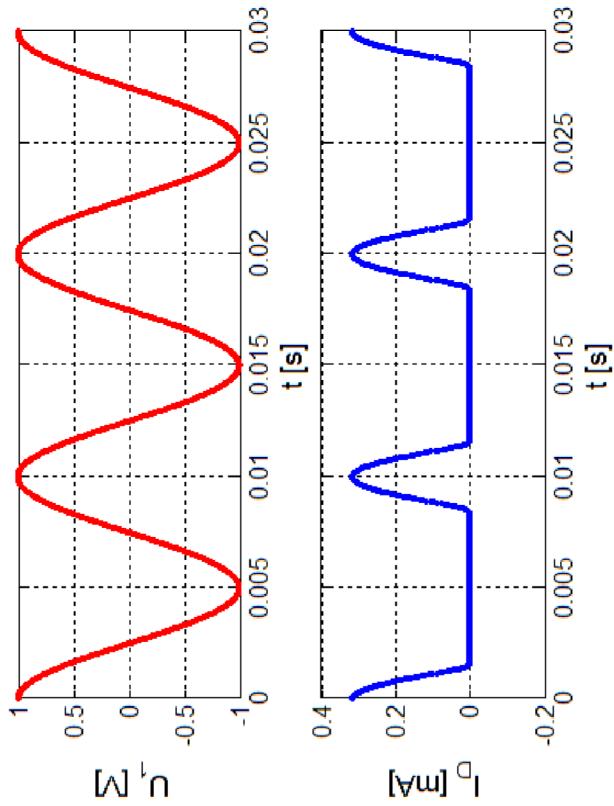
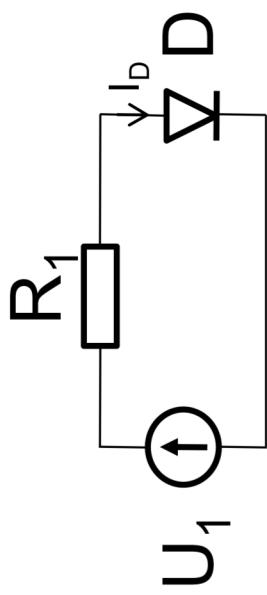
for k=1:length(U1),
Ua=0; Ub=U1(k); Ux=(Ua+Ub)/2; Ierr=100;

while abs(Ierr)>1e-10,
Ix=Is*(exp(Ux/UT)-1);
Ierr=Ix-(U1(k)-Ux)/R1;
if ((Ierr>0)&(U1(k)>=0))|((Ierr<0)&(U1(k)<0)),
Ub=Ux;
else
Ua=Ux;
end;

Ux=(Ua+Ub)/2;

end;
ID(k)=Ix;
end;

```



# Modelowanie tranzystora MOS z kanałem n, wz bogacanym (NMOS)

## Zakres liniowy pracy

$$0 \leq U_{DS} \leq U_{GS} - U_{Th}$$

$$I_D = \frac{W}{L} \cdot \mu_n \cdot C_{ox} \cdot \left[ (U_{GS} - U_{Th}) \cdot U_{DS} - \frac{U_{DS}^2}{2} \right]$$

## Zakres nasycenia

$$0 \leq U_{GS} - U_{Th} \leq U_{DS}$$

$$I_D = \frac{W}{2 \cdot L} \cdot \mu_n \cdot C_{ox} \cdot (U_{GS} - U_{Th})^2 \cdot (1 + \lambda \cdot U_{DS})$$

**Zakres nasycenia**

```

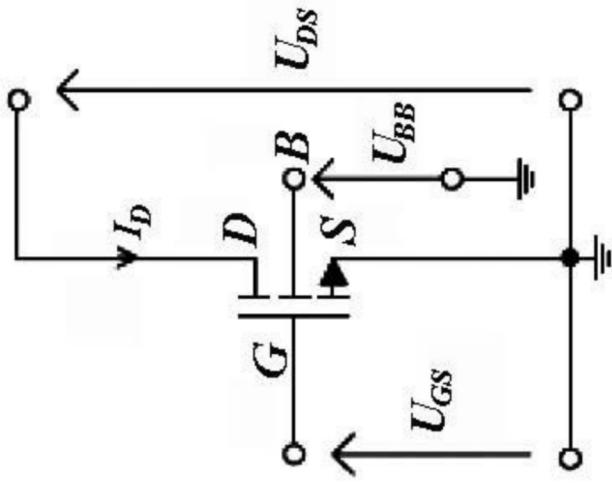
function ID=mosfet_model_1(UGS,UDS);
% ID=mosfet_model_1(UGS,UDS);
%
% Uwaga - obliczenia tylko dla skalarnych danych!

```

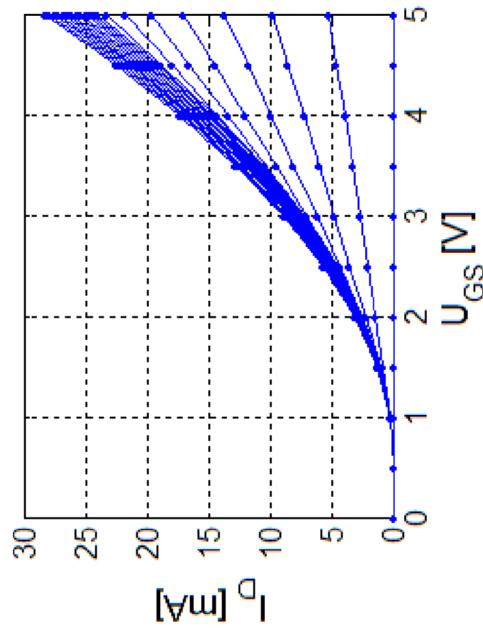
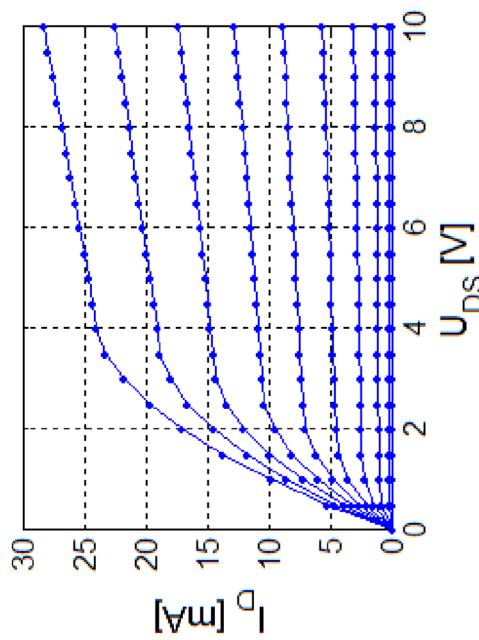
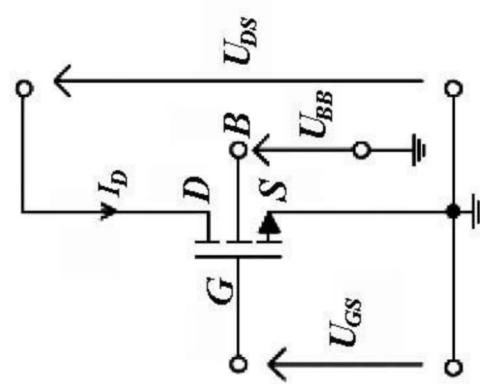
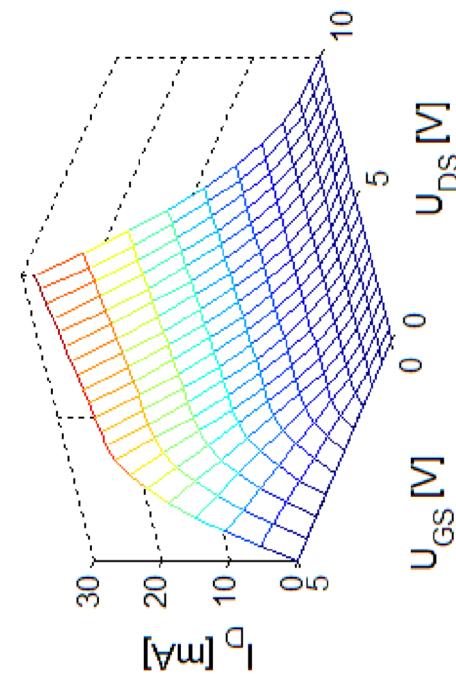
```

unCox=100e-3; % [mA/V^2];
WdoL=25;
lambda=0.03; % [1/V];
UTH=0.6; % [V];
UGST=UGS-UTH;
if UGST>=0,
    if UDS>=UGST,
        ID=WdoL/2*unCox*(UGS-UTH)^2*(1+lambda*(UDS-UGST));
    else
        ID=WdoL*unCox*((UGS-UTH)*UDS-UDS^2/2);
    end;
else
    ID=0;
end;
% KONIEC FUNKCJI;

```



# Charakterystyki tranzystora MOSFET



```

UGS=0:0.5:5; M=length(UGS);
UDS=0:0.5:10; N=length(UDS);
for m=1:M,
    for n=1:N,
        ID(m,n)=mosfet_model_1(UGS(n),UDS(n));
    end;
end;

% Jak to samo zrealizowac macierzowo?
figure(1);
subplot(2,2,1); mesh(UDS,UGS,ID);

```

# Tranzystor bipolarny

```
% m-plik skryptowy: charakterystyki_tranzystora_bipolarnego.m
%
% Tranzystor;
%
clc; clear; close all;

UBE=0:0.01:0.8; N=length(UBE);
UCE=0:0.01:5; M=length(UCE);

for m=1:M,
    for n=1:N,
        [IB(m,n),IC(m,n)]=tranzystor(UBE(n),UCE(m));
    end;
end;

%
% KONIEC FUNKCJI;
```

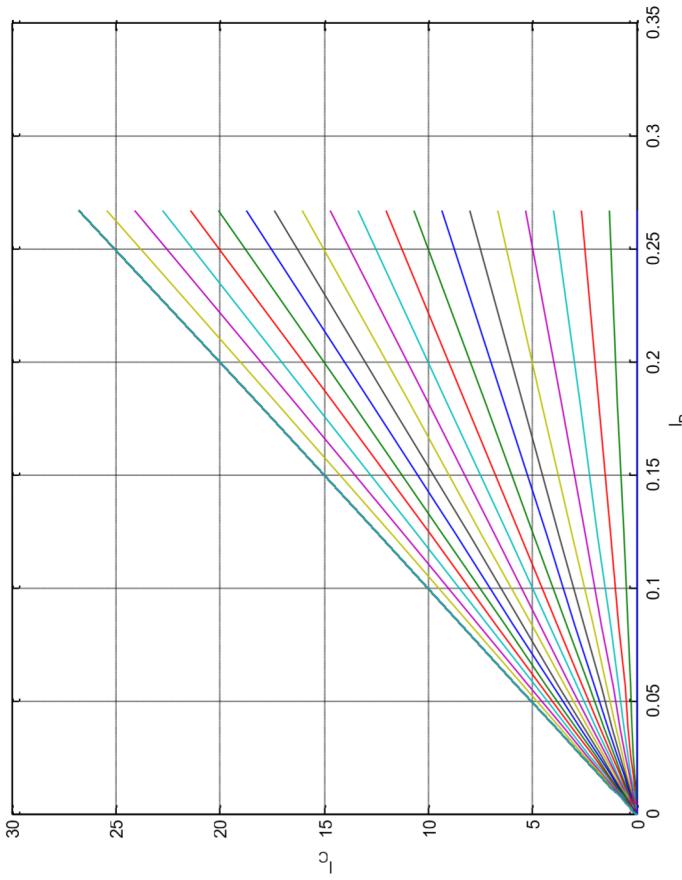
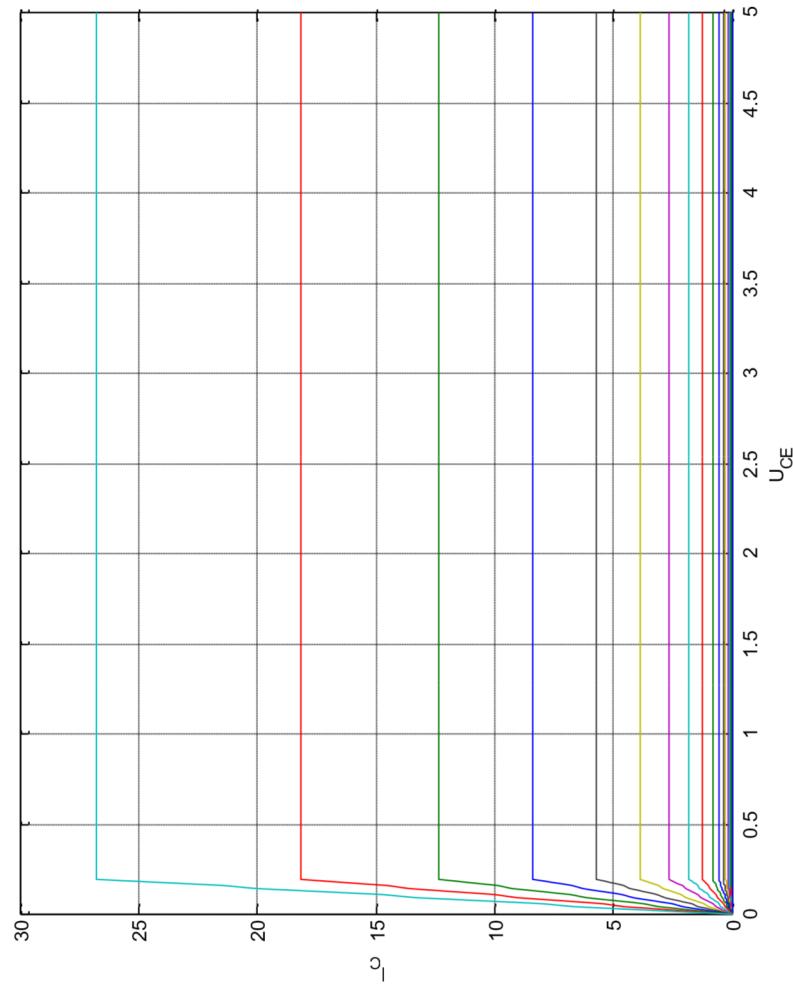
```
function [IB,IC]=tranzystor(UBE,UCE);

B=100; % Beta;
IB=model_diody(UBE);
IC=B*IB;
if (UCE>=0)&(UCE<0.2),
    IC=(UCE/0.2)*IC;
elseif UCE<0,
    IC=0;
end;
```

```
figure(1);
plot(UCE,IC); grid on;
xlabel('U_C_E'); ylabel('I_C');
figure(2);
plot(IB,IC); grid on;
xlabel('I_B'); ylabel('I_C');
figure(3);
plot(UBE,IB,'b-'); grid on;
xlabel('U_B_E'); ylabel('I_B');

%
% KONIEC PLIKU;
```

# Transistor bipolarny



```

function [IB,IC]=tranzystor(UBE,UCE);
B=100; % Beta;
IB=model_diody(UBE);
IC=B*IB;
if (UCE>=0)&(UCE<0.2),
IC=(UCE/0.2)*IC;
elseif UCE<0,
IC=0;
end;
```



# Rozwiązywanie równania metodą optymalizacyjną

$$f(x) = 0 \iff x = ?$$

$$x^2 - 2 \cdot x + 4 = 0$$

$$x^2 - 2 \cdot x - 4 = 0$$

```
% m-plik skryptowy: rozwiazanie_rownania_1.m
```

```
% Autor i kiedy opracował;
```

```
clc; clear; close all;
```

```
x0=-1;  
% x0=1;
```

```
a=[1,-2,-4];  
% a=[1,-2,4];
```

```
x=fminsearch(@zadanie1,x0,[],a),  
test=polyval(a,x),  
roots(a),
```

```
% KONIEC PLIKU;
```

Plik: **zadanie1.m**

```
function F=zadanie1(x,a);  
  
y=polyval(a,x);  
F=abs(y);  
  
% KONIEC FUNKCJI;
```

# Rozwiązywanie równania metodą optymalizacyjną (cd.)

## Układ z diodą – punkt pracy

```
% m-plik skryptowy: punkt_pracy_ukladu.m
```

```
% Autor i kiedy opracował;
```

```
U1=2;
R1=100;
UD0=1e-3;
```

```
UD=fminsearch(@uklad_1,UD0,[],R1,U1),
ID=model_diody(UD),
```

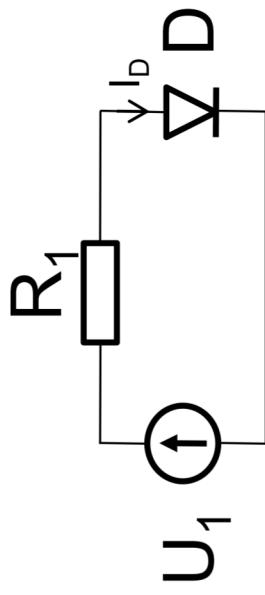
```
% KONIEC PLIKU;
```

```
function I=model_diody(U);
```

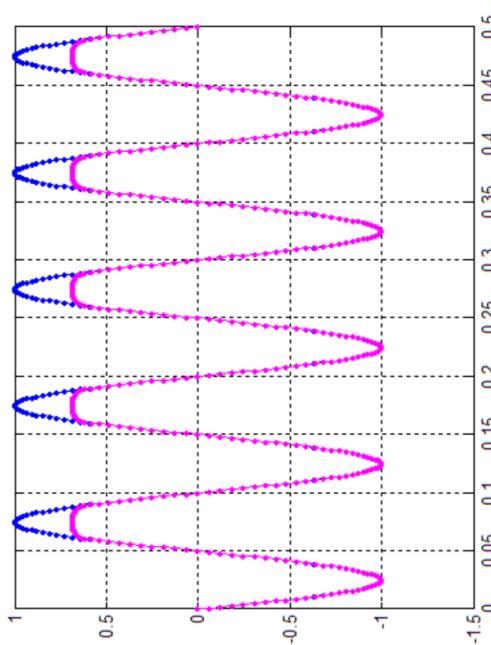
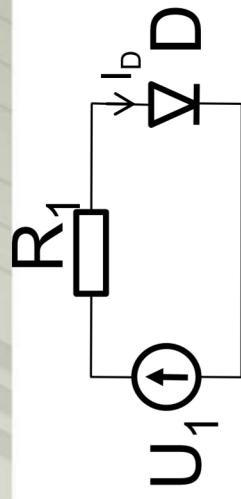
```
q=1.6e-19; % [C];
T=300; % [K];
k=1.38e-23; % stała Boltzmanna [J/K];
UT=k*T/q;
Is=1e-14;
I=Is*(exp(U/UT)-1);
```

```
% KONIEC FUNKCJI;
```

```
function F=uklad_1(UD,R,U);
Ix=(U-UD)/R;
ID=model_diody(UD);
F=(ID-Ix)^2;
% KONIEC FUNKCJI;
```



# Rozwiązywanie równania metodą optymalizacyjną (cd.)



```
% m-plik skryptowy: praca_ukladu_sinus.m
% Autor i kiedy opracował;
clc; clear; close all;

f1=10; T=1/f1; dt=0.01*T; t=0:dt:5*T;
R1=100;
U1=-sin(2*pi*f1*t);
tic;
k=0;
for UX=U1;
    k=k+1;
    UD0=0;
    UD(k)=fminsearch(@uklad_1,UD0,[],R1,UX);
end;
czas_w_minutach=toc/60,
%
```

```
function F=uklad_1(UD,R,U);
Ix=(U-UD)/R;
ID=model_diody(UD);
F=(ID-Ix)^2;
```

```
%
% KONIEC FUNKCJI;
```

# Sprzężenie zwrotnie (ujemne)

```
% m-plik skryptowy: test_sprzezenia_zwrotnego.m
%
% Konfrontacja dwoch interpretacji;
%
% Informacje typu „kto i kiedy”;
%
clc; clear; close all;

h=inline('( $x+1.5$ ).^2','x');
% lub funkcja "anonymowa";
dt=0.001; tmax=2; t=0:dt:tmax; N=length(t);

f1=5; x=cos(2*pi*f1*t); y=h(x);

a=0.5; % współczynnik sprzężenia zwrotnego;

% $y_1(1)=h(x(1));$ 
y1(1)=fminsearch(@systemik1, h(x(1)), [], a, x, 1, h);
tic;
for n=2:N,
    y1(n)=h(x(n)-a*y1(n-1));
end; czas1_obliczen=toc,

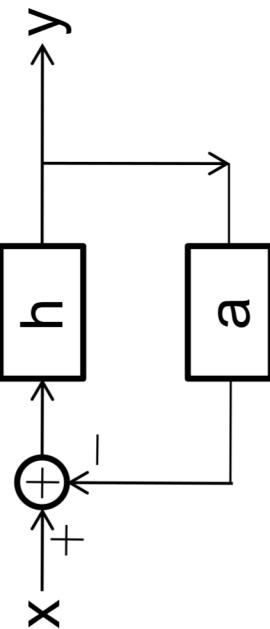
y2(1)=fminsearch(@systemik1, h(x(1)),[], a, x, 1, h);
tic;
for n=2:N,
    y2(n)=fminsearch(@systemik1, y2(n-1), [], a, x, n, h);
end; czas2_obliczen=toc,
```

**Dokończenie m-pliku:**

```
figure(1); clf;
subplot(3,1,1); plot(t,x,'b.-'); grid on;
subplot(3,1,2); plot(t,y,'r.-'); grid on;
subplot(3,1,3); plot(t,y1,'m.-'); grid on; hold on;
plot(t,y2,'k.-');
```

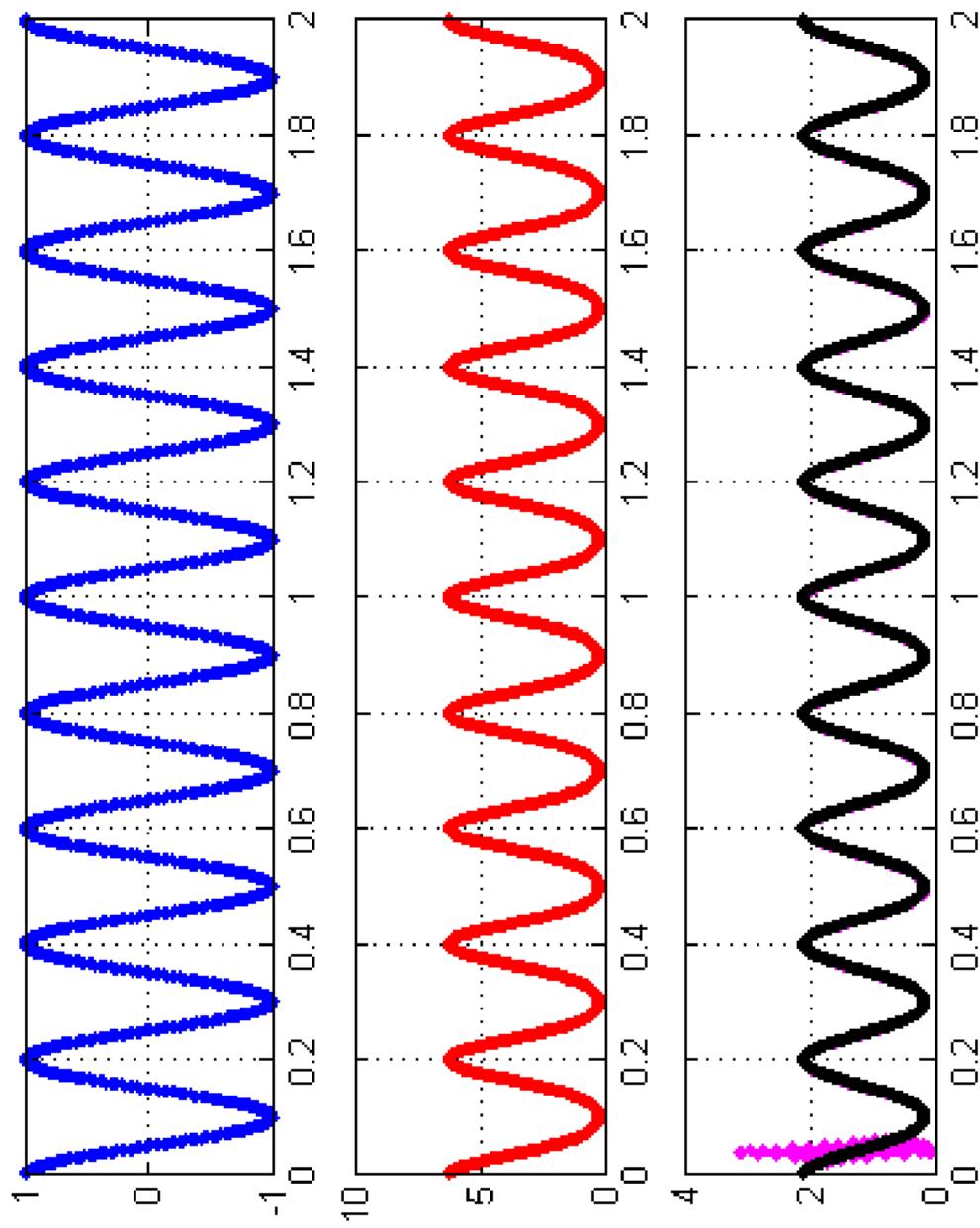
**Dokończenie m-pliku:**

```
function F=systemik1(y,a,x,n,h);
y_ref=h(x(n)-a*y);
err=y-y_ref;
F=sum(err.^2);
% KONIEC FUNKCJI;
```



# Sprzężenie zwrotne (ujemne)

Wynik uruchomienia:



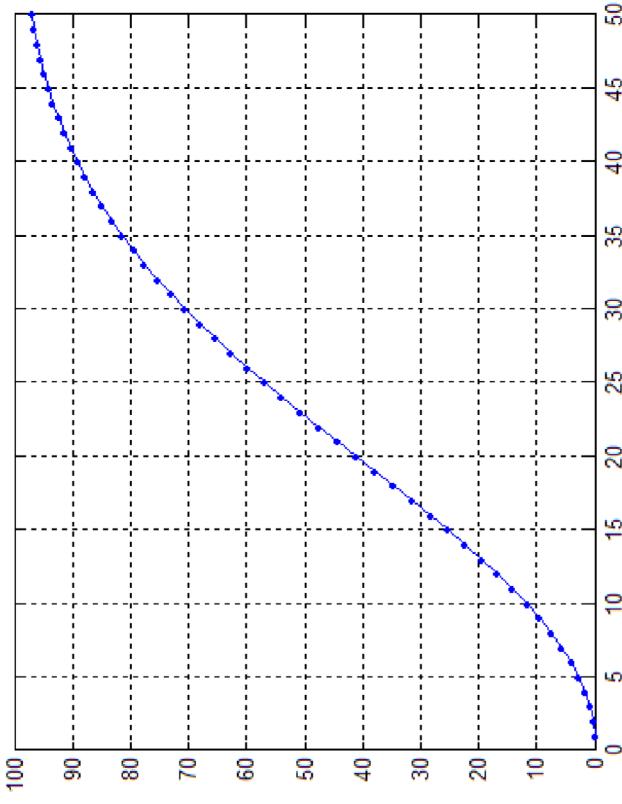
# Jakie jest prawdopodobieństwo zdarzenia, że co najmniej dwie osoby z klasą urodziły się tego samego dnia?

.....  
D=365; % liczba dni w roku;

```
N=50; % maksymalna licznośc klasy;  
for n=1:50, % liczba osób w klasie...  
% liczba n-elementowych wariacji z  
% powtóżeniami zbioru D-elementowego:  
W1=D^n;  
% liczba n-elementowych wariacji bez  
% powtóżen zbioru D-elementowego:  
W2=D;  
for k=2:n,  
W2=W2*(D-k+1);  
end;  
P(n)=1-W2/W1;  
end;
```

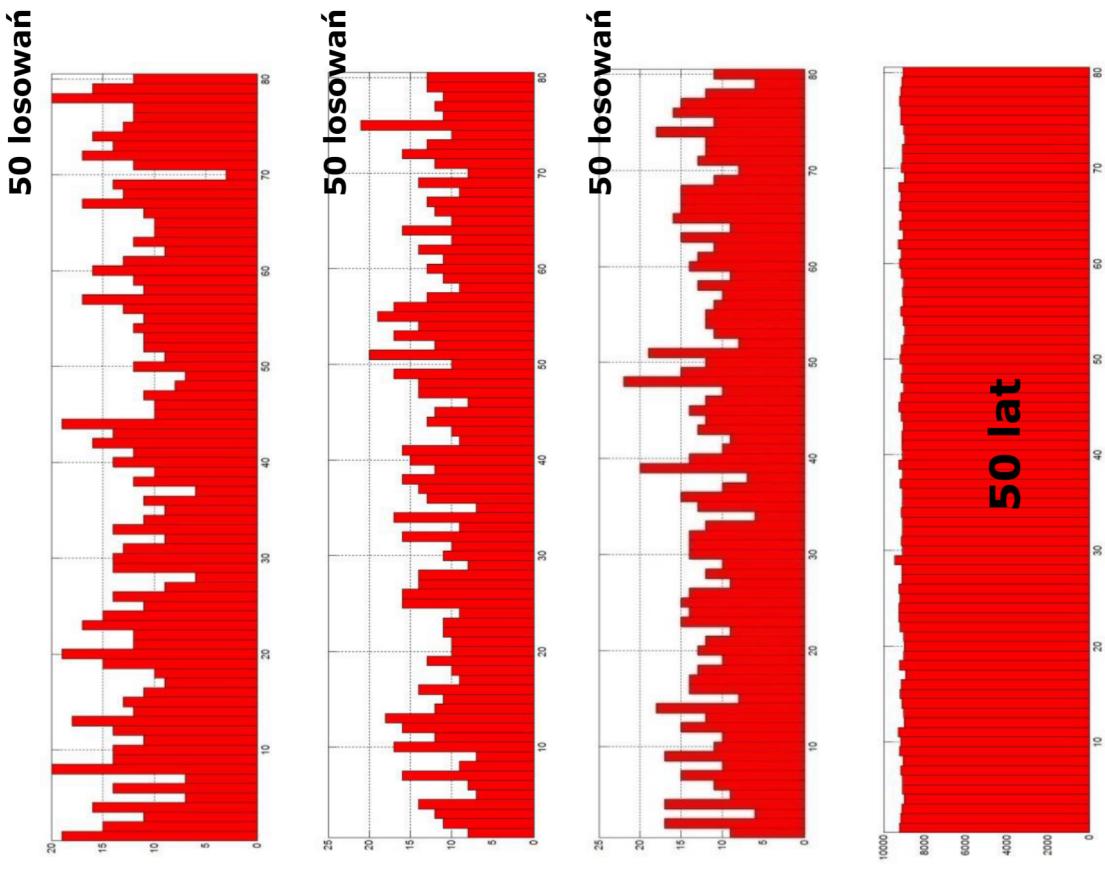
```
figure(1);  
plot(1:N,100*P,'b.-'); grid on;  
.....
```

%



Liczba osób w klasie

# Multi Multi (dawniej: Multilotek)



**Typujemy liczby naturalne od 1 do 80.**

**Koszt pojedyńczego zakładu: 2,50 zł.  
W każdym losowaniu otrzymuje się  
20 liczb.  
Dwa losowania dziennie.**

	10	9	8	7	6	5	4	3	2	1
10	250 000 zł									
9	10 000 zł	70 000 zł								
8	520 zł	2 000 zł	22 000 zł							
7	140 zł	300 zł	600 zł	6 000 zł						
6	12 zł	42 zł	60 zł	200 zł	1 300 zł					
5	4 zł	8 zł	20 zł	20 zł	120 zł	700 zł				
4	2 zł	2 zł	4 zł	4 zł	8 zł	20 zł	84 zł			
3					2 zł	2 zł	4 zł	8 zł	54 zł	
2								2 zł	2 zł	16 zł
1										4 zł
0										

*Zapraszam do laboratorium ...*