

GENERACJA PRZEBIEGU SINUSOIDALNEGO.

Podstawą generacji sygnału sinusoidalnego jest równanie różnicowe wyprowadzone w sposób następujący.

Transmitancja układu generującego jest równa:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{Z[\sin(\omega n)]}{Z[\delta(n)]}$$

Na wyjściu spodziewany jest sygnał sinusoidalny.

Jego transformata jest równa:

$$Y(z) = \frac{C * z^{-1}}{1 + A * z^{-1} + z^{-2}}$$

gdzie:

$$A = 2 * \cos(2 * \pi / N)$$

$$C = \sin(2 * \pi / N)$$

N – liczba próbek na okres sinusoidy.

Na wejście podawany jest sygnał jednostkowy i jego transformata jest równa 1

$$X(z) = 1$$

Z powyższego wynika:

$$\frac{Y(z)}{X(z)} = \frac{C * z^{-1}}{1 + A * z^{-1} + z^{-2}}$$

Stąd po wymnożeniu stronami i zastosowaniu odwrotnej transformaty Z otrzymujemy

$$y(n) = C * x(n-1) + A * y(n-1) - y(n-2)$$

Założmy, że układ zostaje pobudzony impulsem jednostkowym $x(0) = 1$.

Rozważmy równanie różnicowe:

$$\text{gdy } n = 0 \text{ to } y(0) = C * x(-1) + A * y(-1) - y(-2) = 0$$

$$\text{gdy } n = 1 \text{ to } y(1) = C * x(0) + A * y(0) - y(-1) = C$$

$$\text{gdy } n = 2 \text{ to } y(2) = C * x(1) + A * y(1) - y(0) = A * y(1)$$

$$\text{gdy } n = 3 \text{ to } y(3) = C * x(2) + A * y(2) - y(1) = A * y(2) - y(1)$$

...

$$\text{gdy } n = k \text{ to } y(k) = C * x(k-1) + A * y(k-1) - y(k-2) = A * y(k-1) - y(k-2)$$

Stąd równanie różnicowe przybiera postać:

$$y(0) = 0$$

$$y(1) = C$$

$$y(k) = A * y(k-1) - y(k-2)$$

Aby wyznaczyć kolejny wyraz tego ciągu wystarczy mieć dwa poprzednie i można to zapisać w postaci programu (:= oznacza instrukcję podstawienia):

$$y[1] := C;$$

<<<warunki początkowe

$$y[0] := y[2] := 0;$$

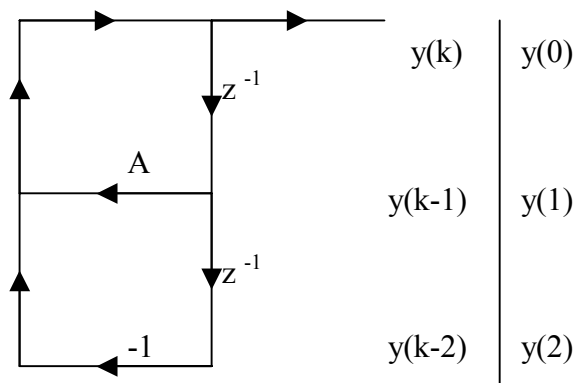
<<<warunki początkowe

$$y[2] := y[1];$$

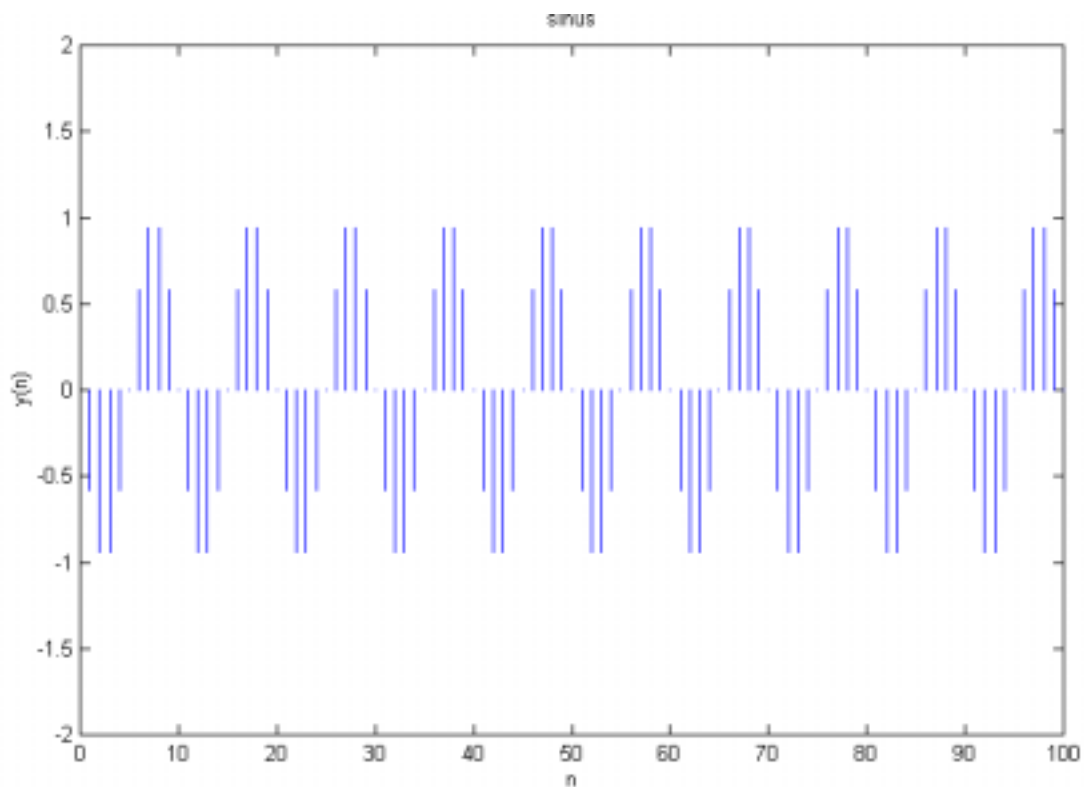
<<<blok operacji powtarzanych

$$y[1] := y[0];$$

$$y[0] := A * y[1] - y[2];$$



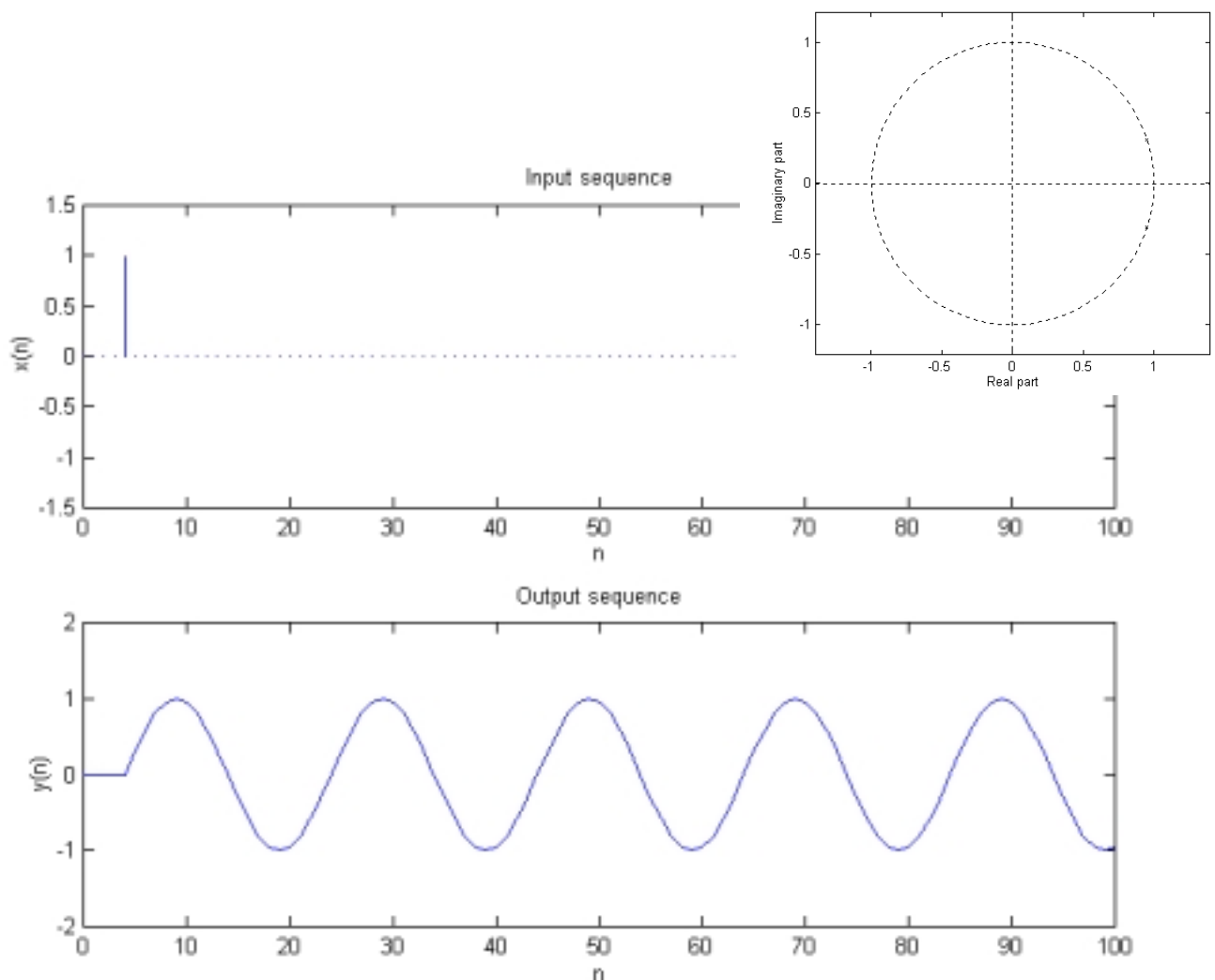
```
% singen.m: Matlab sinus generator
%
close;
n=1:10000;
N=10; % liczba próbek na okres sinusoidy
a=0.9;
A=2*cos(2*pi/N);
C=sin(2*pi/N);
r=zeros(1,100); % tablica próbek
y=[0,C,0];
for j=n,
    y(3)=y(2); % procedura generacji
    y(2)=y(1);
    y(1)=A*y(2)-y(3);
    r(j)=y(1);
end;
grid on;
subplot(1,1,1);stem(n,r)
axis([0,100,-2,2])
title('sinus');xlabel('n');ylabel('y(n)')
pause;
sound(r);
close;
```



```

% sintry.m : Sinus try - z równania różniczkowego
% diff eqn:  $y(n) - A y(n-1) + y(n-2) = C x(n-1)$ 
N=20;
b = [0, sin(2*pi/N)]; a = [1, -2*cos(2*pi/N), 1];
n=0:100;
x = [(n==4)]; %delta
y = filter(b,a,x);
subplot(2,1,1); stem(n,x); title('Input sequence')
xlabel('n'); ylabel('x(n)'); axis([0,100,-1.5,1.5])
subplot(2,1,2); plot(n,y); title('Output sequence')
xlabel('n'); ylabel('y(n)'); axis([0,100,-2,2])
pause;
%xic=filtic(b,a,[0],[sin(2*pi/N)]) %filt and initial condition: y(-1) x(-1)
%xic=filtic(b,a,[0],[10]) %filt and initial condition: y(-1) x(-1)
xic=filtic(b,a,[1]) %filt and initial condition: y(-1)
x=zeros(1,101); %brak sygnału wejściowego
y = filter(b,a,x,xic);
subplot(2,1,1); stem(n,x); title('Input sequence')
xlabel('n'); ylabel('x(n)'); axis([0,100,-1.5,1.5])
subplot(2,1,2); plot(n,y); title('Output with initial cond.')
xlabel('n'); ylabel('y(n)'); axis([0,100,-1.2*abs(max(y)),1.2*abs(max(y))])
pause;
close;
zplane(b,a);

```



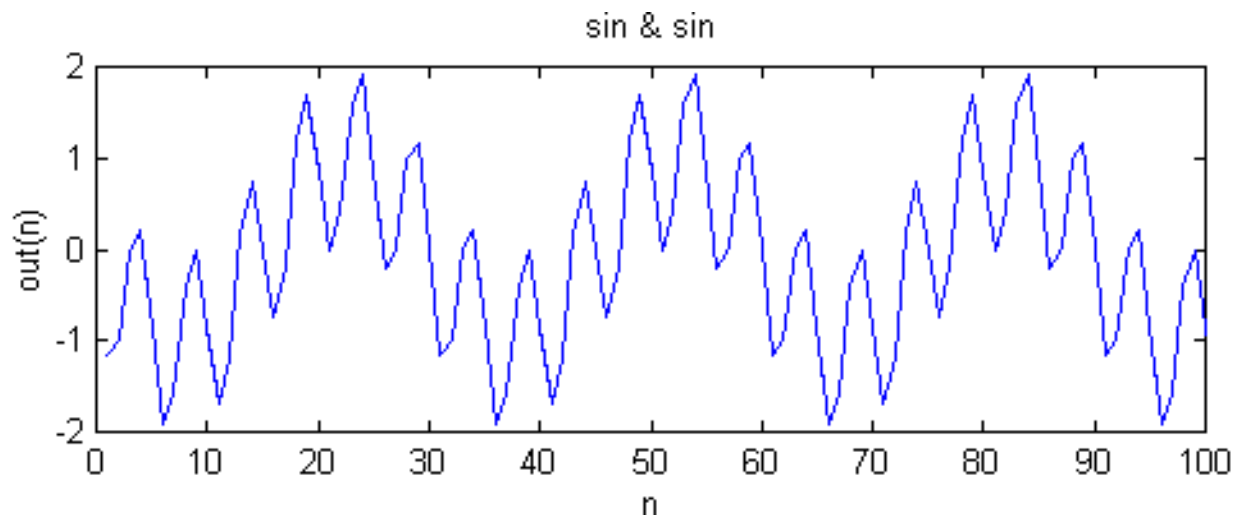
```

% sin2gen.m : Generacja 2 funkcji sinus. Wybieranie tonowe DTMF
%
%
%           Column
%           1       2       3       4
%           1209Hz  1336Hz  1477Hz  1633Hz
%
%           +-----+-----+-----+
%           | 1 | 2 | 3 | A | 697Hz
%           |   |   |   |   |
%           +-----+-----+-----+
%           | 4 | 5 | 6 | B | 770Hz
%           |   |   |   |   |
%           +-----+-----+-----+
%           | 7 | 8 | 9 | C | 852Hz
%           |   |   |   |   |
%           +-----+-----+-----+
%           | * | 0 | # | D | 941Hz
%           |   |   |   |   |
%           +-----+-----+-----+
%
close;
n=1:10000;
N1=5;
A1=2*cos(2*pi/N1);
C1=sin(2*pi/N1);
y1=[0,C1,0];

N2=30;
A2=2*cos(2*pi/N2);
C2=sin(2*pi/N2);
y2=[0,C2,0];

for j=n,
    y1(3)=y1(2);
    y1(2)=y1(1);
    y1(1)=A1*y1(2)-y1(3);
    y2(3)=y2(2);
    y2(2)=y2(1);
    y2(1)=A2*y2(2)-y2(3);
    r(j)=y1(1)+ y2(1);
end;
grid on;
subplot(2,1,1);plot(n,r)
axis([0,100,-2,2])
title('sin & sin');xlabel('n');ylabel('out(n)')
pause;
sound(r);
close;

```



```

% GOERTZEL.m - badanie algorytmu
clc;
echo on;
% ALGORYTM GOERTZEL'A
% Prezentacja algorytmu Goertzel'a
% Implementacja algorytmu znajduje się w funkcji amp2= zgoer(x, N, k)
N=200; %liczba próbek sygnału x
fp=8000;

echo off;

%f= 350.7;
%t= [1:N]; % N próbek
%x= sin(2*pi*(f/fp) * t)+0.2*randn(size(t)); % sinus_szum

f1= 1335;
f2= 851.2;
t= [1:N]; % N próbek
x= sin(2*pi*(f1/fp) * t) + sin(2*pi*(f2/fp) * t); % sinus+sinus

%t= [1:N]; % N próbek % chirp
%x= sin(2*pi*(t/fp) .* t);

subplot(3,1,1);
plot(x);

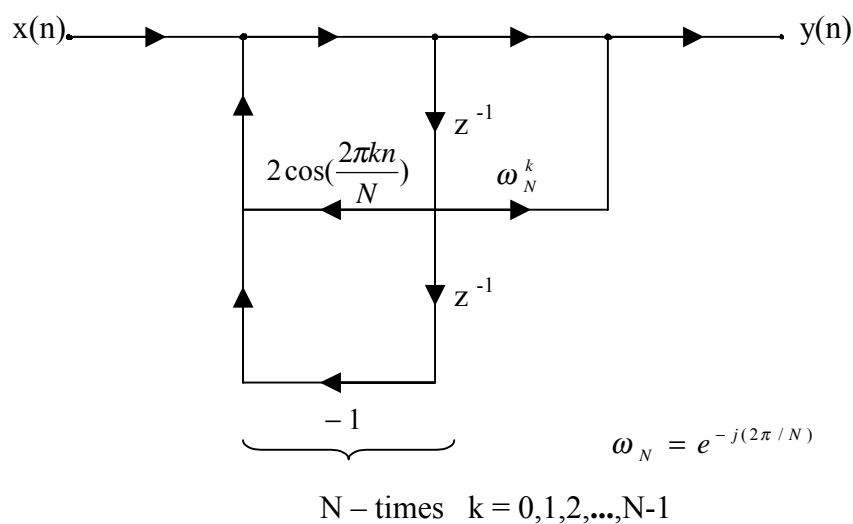
amp2= zeros(1, N);
for i= 0:N-1,
    amp2(i+1)= zgoer(x, N, i);
end;

subplot(3,1,2);
plot(sqrt(amp2)); % algorytm Goertzel'a

subplot(3,1,3);
plot(abs(fft(x,N)), 'r') % dla porównania DFT

% ALGORYTM GOERTZEL'A
% Obliczanie kwadratu amplitudy dla k-tej harmonicznej
% N-elementowego zbioru danych x.
function amp2= zgoer(x,N,k)
% zerowanie 2-elementowej historii w stanie początkowym
hist= zeros(1,2);
% wstępne obliczenie współczynnika fact
factor=2*cos(2*pi*k/N);
% główna pętla filtru dla N-elementowego zbioru x
for i=1:N,
    hist=[x(i)+factor*hist(1)-hist(2),hist(1)];
end;
% obliczenie kwadratu amplitudy po N obrotach filtru
% korzystając z zawartości historii
amp2=hist(1)*hist(1)+hist(2)*hist(2)-factor*hist(1)*hist(2);

```



$$H_k(z) = \frac{1 - \omega_N^k z^{-1}}{1 - 2 \cos(2\pi k / N) z^{-1} + z^{-2}}$$

