

Przetwarzanie Sygnałów

Jakub Dudarewicz

Semestr letni, 2016

Zadanie 1. Instrukcja 0

Zadanie 1.1 Operacje tablicowe i macierzowe

Wynik operacji mnożenia macierzowego i tablicowego:

A =				tablicowe			
1	43	45		6	516	2025	
				2	172	2520	
B =				12	129	45	
6	12	45					
2	4	56		macierzowe			
12	3	1		632	319	2498	

Operacje te są fundamentalnie od siebie różne

Zadanie 1.2 Próbkowanie przebiegu sinusoidalnego

M-plik użyty do generacji wykresów:

```
% probkowanie sinusa
n=10; %długość obserwacji
k=100; %liczba próbek
t=[1:k];

H = figure;

m=500;
a=sin(2*pi*n*t/m);
subplot(221);

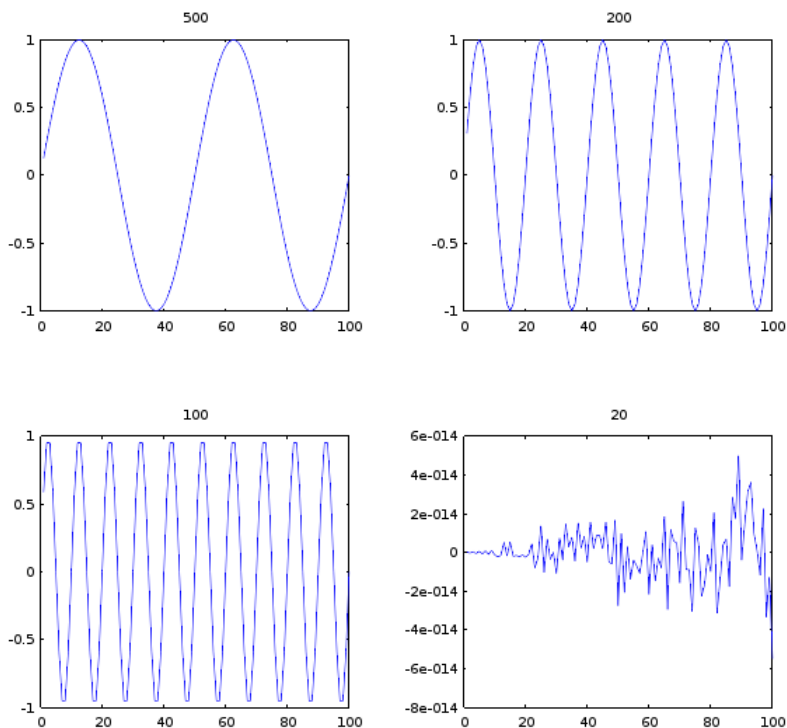
plot(a);
title("500");

m=200;
a=sin(2*pi*n*t/m);
subplot(222);
plot(a);
title("200");

m=100;
a=sin(2*pi*n*t/m);
subplot(223);
plot(a);
title("100");

m=20;
a=sin(2*pi*n*t/m);
subplot(224);
plot(a);
title("20");

clear;
```



Rysunek 1: Przebieg sinusoidalny próbkowany z różnymi częstotliwościami

Liczba zarejestrowanych okresów i rozdzielczość wynika z częstotliwości próbkowania. W tym zadaniu nie przekształcam dziedziny sygnału do czasu.

Zadanie 1.3 Skok jednostkowy

M-plik użyty do generacji wykresów:

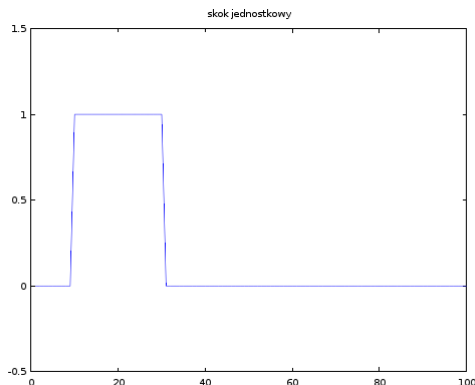
```
%skok jednostkowy

be = 10;
en = 30;
dur = 100;

a = zeros(dur,1);

for i = be:en
    a(i, 1) = 1;
    be = be + 1;
end

figure;
plot(a);
axis([0, 100, -0.5, 1.5]);
title("skok jednostkowy");
```



Rysunek 2: Przykładowy wydruk skoku jednostkowego

Zadanie 1.4 Skok jednostkowy, użycie varargin

M-plik użyty do generowania wykresów:

```
function skok(varargin) %(duration,
begin 1, end 1, begin 2, end 2...)

k = 2;
l=1;
figure;
r = ceil((nargin-1)/4);
dur = varargin{1};

for i = 1:((nargin-1)/2)
    be = varargin{k};
    en = varargin{k+1};

    a = zeros(dur,1);

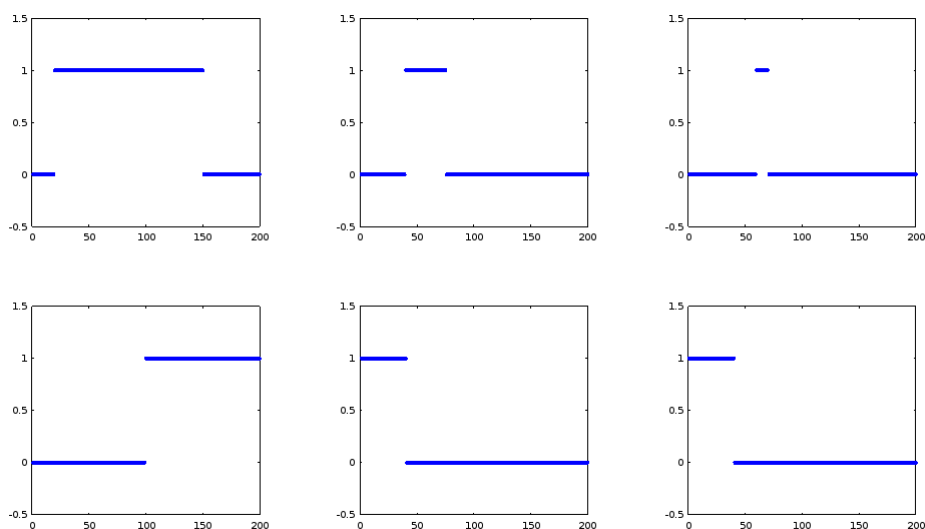
    for j = be:en
        a(j, 1) = 1;
        be += 1;
    end

    subplot(r, 2, 1);
    plot(a, '.');
    axis([0, dur, -0.5, 1.5]);

    printf("bound %d - %d\n", varargin{k},
varargin{k+1});

    if(k+1<(nargin-1))
        k=k+2;
    endif
    l=l+1;
end

endfunction
```



Rysunek 3: Przykładowy wydruk dla siedmiu argumentów

Funkcja varargin i wszystkie jej towarzyszące są bardzo użyteczne, bo pozwalają na napisanie bardziej uniwersalnej funkcji.

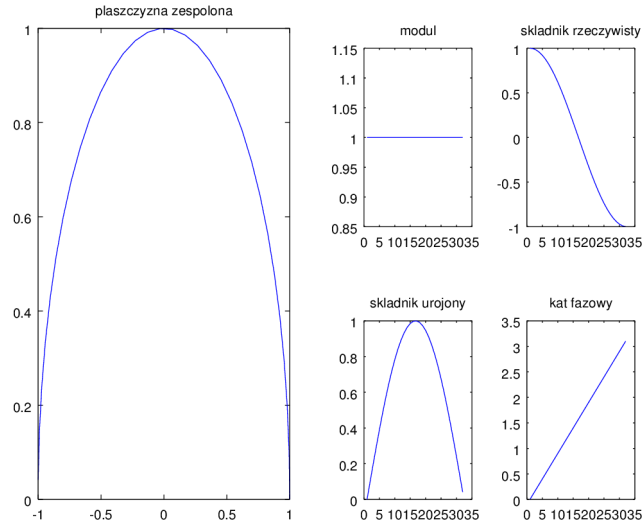
Zadanie 1.5 Przebieg wykładniczy zespolony

M-plik użyty do generacji wykresów funkcji $f(x) = e^{xi}$:

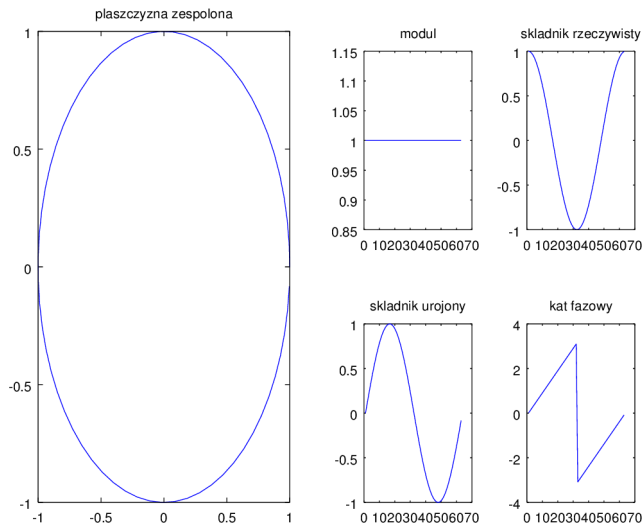
```
%przebieg wykładniczy zespolony
t=[0:0.1:2*pi];
zesp=e.^(t.*j);
rzecz=real(zesp);
uroj=imag(zesp);
modul=abs(zesp);
faza=angle(zesp);

figure
subplot(121)
plot(zesp)
title("plaszczyna zespolona")
subplot(243)
plot(modul)
title("modul")

subplot(244)
plot(rzecz)
title("skladnik rzeczywisty")
subplot(247)
plot(uroj)
title("skladnik urojony")
subplot(248)
plot(faza)
title("kat fazowy")
```



(a) $0i - \pi i$



(b) $0i - 2\pi i$

Rysunek 4: Zespolone przebiegi wykładnicze dla dwóch zakresów zmiennej zespolonej

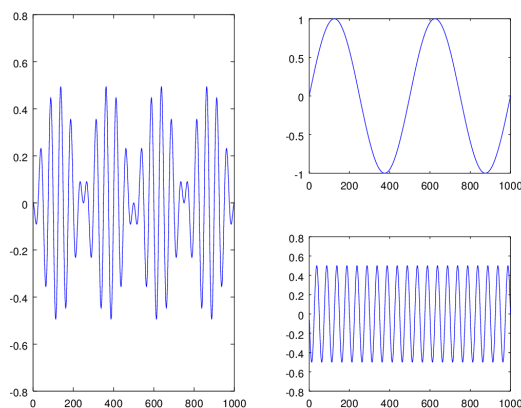
Z powyższych wykresów połączenie funkcji wykładniczej zespolonej z funkcjami trygonometrycznymi jest oczywiste. Zrozumiała też staje się słynna Tożsamość Eulera $e^{\pi i} + 1 = 0$.

Zadanie 1.6 Modulowanie przebiegu sinusoidalnego

M-plik użyty do generacji wykresów:

```
%modulowanie sinusa  
  
n=1; %długość obserwacji  
k=1000; %liczba próbek  
t=1:k; %probki  
m=500; %czestotliwosc obserwacji  
  
a=sin(2*pi*n*t/m);  
b=(0.5).sin((10*2*pi*n*t/m)+pi);
```

```
c=a.*b;  
  
figure  
  
subplot(121)  
plot(c)  
subplot(222)  
plot(a)  
subplot(224)  
plot(b)
```



Rysunek 5: Z lewej - sygnał zmodulowany, z prawej - sygnały wyjściowe

Modulacja osiągnana jest poprzez pomnożenie tablicowe przez siebie dwóch sygnałów.

Zadanie 2. Instrukcja 1

Zadanie 2.1 Generowanie przebiegów sinusoidalnych o pulsacji znormalizowanej

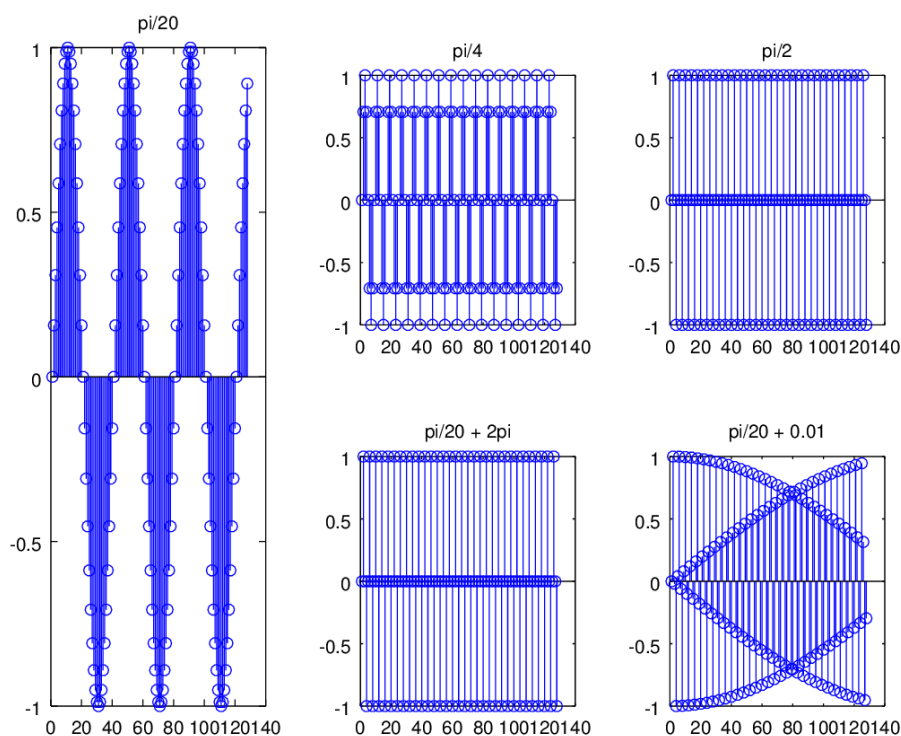
M-plik użyty do generacji wykresów:

```
n=[0:127];

xa=sin((pi/20)*n);
xb=sin((pi/4)*n);
xc=sin((pi/2)*n);
xd=sin((pi/2+2*pi)*n);
xe=sin((pi/2+0.01)*n);

subplot(131)
stem(xa)
title('\pi/20')
subplot(232)

stem(xb)
title('\pi/4')
subplot(233)
stem(xc)
title('\pi/2')
subplot(235)
stem(xd)
title('\pi/2 + 2\pi')
subplot(236)
stem(xe)
title('\pi/2 + 0.01')
```



Rysunek 6: Przebiegi generowane dla różnych pulsacji.

Zbyt duża częstotliwość powoduje niedokładne rejestrowanie przebiegu.

Zadanie 2.2 Generowanie przebiegów sinusoidalnych próbkowanych z różną częstotliwością

M-plik użyty do generacji wykresów:

```
%sampling frequency
fsa=4000;
fsb=4020;
fsc=8000;
fsd=19000;
fse=44000;

%samples vector
n=[0:127];

%signal frequency
f=2000;

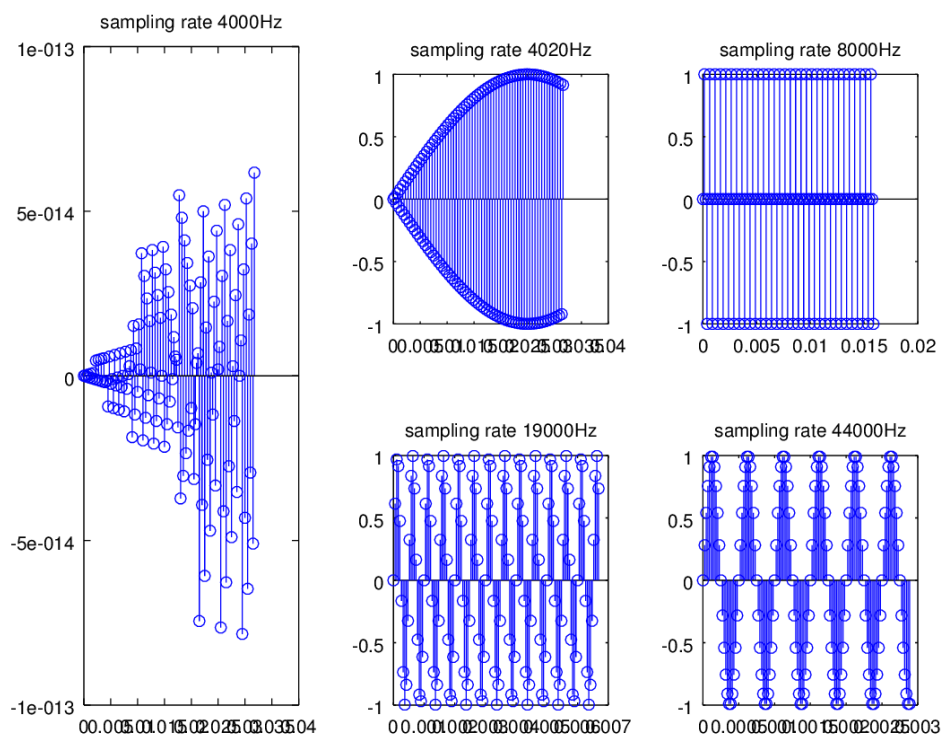
%time vectors
ta=n/fsa;

tb=n/fsb;
tc=n/fsc;
td=n/fsd;
te=n/fse;

%signal model
xa=sin(2*pi*f*ta);
xb=sin(2*pi*f*tb);
xc=sin(2*pi*f*tc);
xd=sin(2*pi*f*td);
xe=sin(2*pi*f*te);

subplot(131)
stem(ta, xa)

title('sampling rate 4000Hz')
subplot(232)
stem(tb, xb)
title('sampling rate 4020Hz')
subplot(233)
stem(tc, xc)
title('sampling rate 8000Hz')
subplot(235)
stem(td, xd)
title('sampling rate 19000Hz')
subplot(236)
stem(te, xe)
title('sampling rate 44000Hz')
```



Rysunek 7: Przebiegi o równej częstotliwości próbkowane z różną częstotliwością wyświetlone w dziedzinie czasu

Zadanie 2.3 Generowanie przebiegów sinusoidalnych przesuniętych w fazie

M-plik użyty do generacji wykresów:

```
%sampling frequency
fs=4000;

%samples vector
n=[0:63];

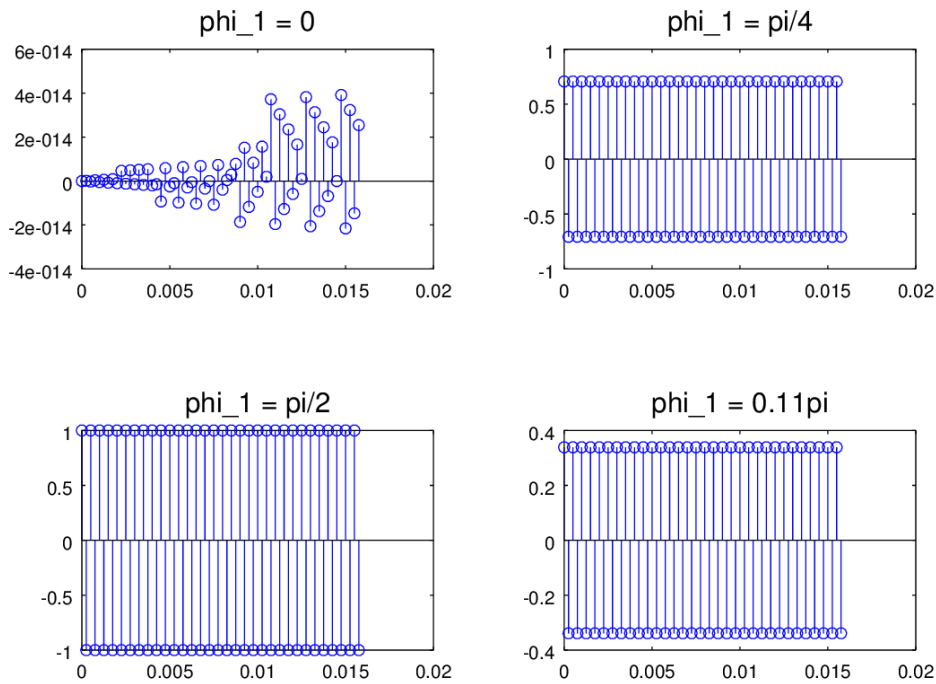
%signal frequency
f=2000;

%time vectors
t=n/fs;

%signal model
xa=sin(2*pi*f*t);
xb=sin(2*pi*f*t + pi/4);
xc=sin(2*pi*f*t + pi/2);
xd=sin(2*pi*f*t + 0.11*pi);

subplot(221)
stem(t, xa)

title('\phi_1 = 0', 'fontsize', 15)
subplot(222)
stem(t, xb)
title('\phi_1 = \pi/4', 'fontsize', 15)
subplot(223)
stem(t, xc)
title('\phi_1 = \pi/2', 'fontsize', 15)
subplot(224)
stem(t, xd)
title('\phi_1 = 0.11\pi', 'fontsize', 15)
```



Rysunek 8: Przebiegi o równej częstotliwości z różnym przesunięciem fazowym i próbkowane z równą częstotliwością

Ponieważ częstotliwość próbkowania sygnału jest dwukrotnie większa od częstotliwości sygnału sygnał jest próbkowany zawsze w tych samych dwóch miejscach w okresie. Od przesunięcia fazowego sygnału zależy jakie wielkości osiągną próbki.

Zadanie 2.4 Generowanie przebiegów sinusoidalnych o różnych częstotliwościach

M-plik użyty do generacji wykresów:

```
%sampling frequency
fs=4000;

%samples vector
n=[0:127];

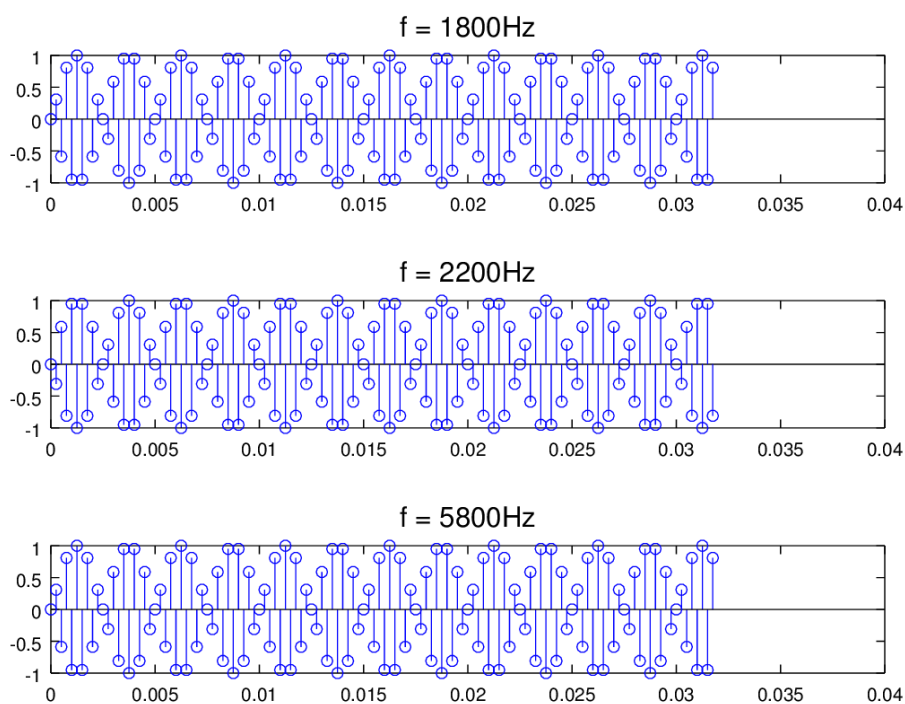
%signal frequency
fa=1800;
fb=2200;
fc=5800;

%time vectors
t=n/fs;

%signal model
xa=sin(2*pi*fa*t);
xb=sin(2*pi*fb*t);
xc=sin(2*pi*fc*t);

subplot(311)

stem(t, xa)
title('f = 1800Hz', 'fontsize', 15)
subplot(312)
stem(t, xb)
title('f = 2200Hz', 'fontsize', 15)
subplot(313)
stem(t, xc)
title('f = 5800Hz', 'fontsize', 15)
```



Rysunek 9: Przebiegi o różnej częstotliwości próbkowane z równą częstotliwością wyświetlone w dziedzinie czasu

Zadanie 2.5 Rekonstrukcja sygnałów cyfrowych

M-plik użyty do generacji wykresów:

```
%sampling frequency
fsa=44000;
fsd=1000;

%samples vector
nd=[0:15];
na=[0:44*length(nd)];

%signal frequency
f=100;

%time vectors
ta=na/fsa;
td=nd/fsd;

%signal model
xa=sin(2*pi*f*ta);
xd=sin(2*pi*f*td);

%reconstructions
xl=interp1(td, xd, ta, "linear");
xn=interp1(td, xd, ta, "nearest");

xs=zeros(size(ta));
for k = 1:length(td);
    st = xd(k)*sinc(fsd*(ta-td(k)));
    xs = xs + st;
end

hold on

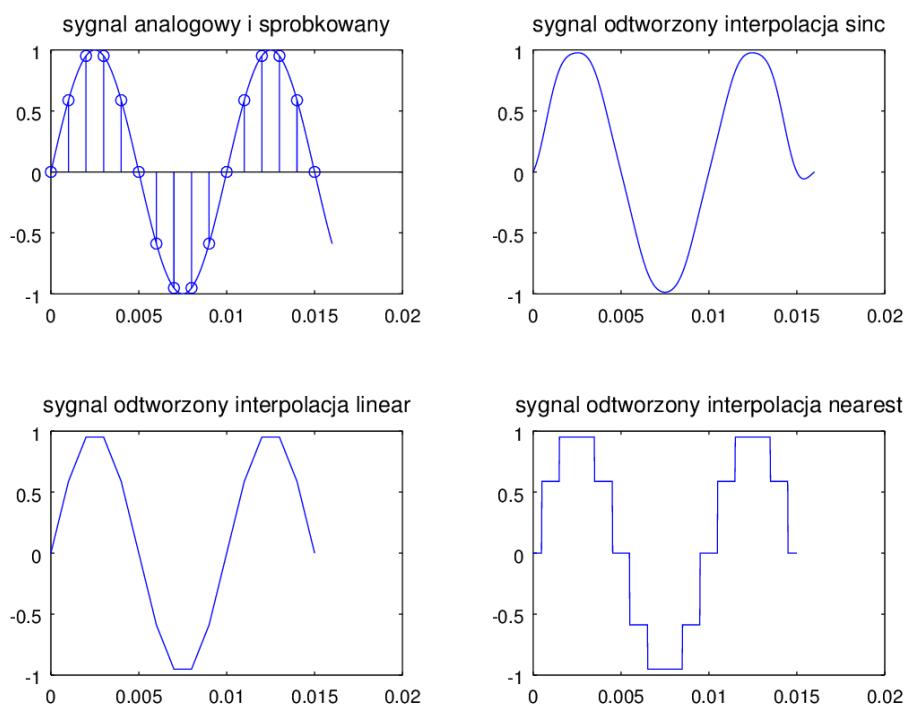
subplot(222)
plot(ta, xs, 'r')
hold on
plot(ta, xa)
title("sygnał odtworzony interpolacja sinc"

, "fontSize", 12)

subplot(223)
plot(ta, xl)
title("sygnał odtworzony interpolacja linear"
, "fontSize", 12)

subplot(224)
plot(ta, xn)
title("sygnał odtworzony interpolacja nearest"
, "fontSize", 12)

subplot(221)
stem(td, xd)
hold on
plot(ta, xa)
title("sygnał analogowy i sprobkowany"
, "fontSize", 12)
```



Rysunek 10: Sygnały zrekonstruowane za pomocą trzech algorytmów