Przetwarzanie sygnałów

Ćwiczenie 2

Podstawowe operacje na sygnałach cyfrowych

dr hab. inż. Tomasz Piasecki (tomasz.piasecki@pwr.edu.pl)

Spis treści

Katedra Nanometrologii
Wydział Elektroniki, Fotoniki i Mikrosystemów
Politechnika Wrocławska

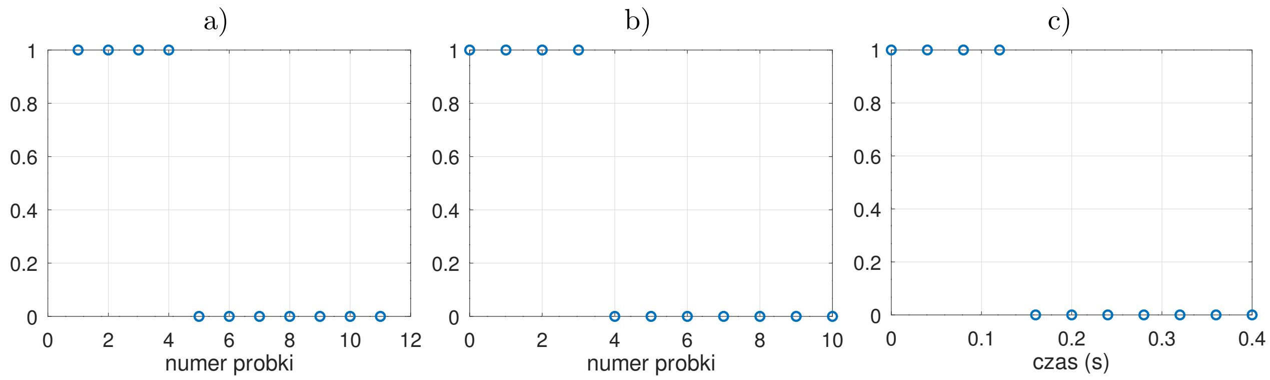
1 Wstęp

1.1 Próbki sygnałów dyskretnych

Sygnał dyskretny zasadniczo przedstawiamy w postaci ciągu wartości, które można zapisać między innymi w jednowymiarowej tablicy. Inną nazwą takich danych to wektor. Na przykład 11 kolejnych próbek impulsu prostokątnego, którego czas trwania to 4 próbki, można zapisać w 11-elementowym wektorze a ten przechowywać w tablicy `x` używając następującej instrukcji Octave:

```
x = [1 1 1 1 0 0 0 0 0 0 0];
```

Chcąc wykreślić przebieg tego wykresu w najprostszy sposób za pomocą instrukcji[1] `plot (x, "o");` otrzyma się wykres nie do końca zgodny z oczekiwaniami (rysunek 1a). Próbka początkowa wykreślona zostanie na współrzędnej poziomej 1 zamiast 0. Wynika to ze sposobu indeksowania tablic w Octave.



Rysunek 1: Przykłady wykresów ilustrujących przebieg sygnału dyskretnego: a) uzyskany wprost z tabeli wartości z niewłaściwymi współrzędnymi poziomymi, b) z numerami próbek na osi poziomej, c) z czasem na osi poziomej.

Chcąc skorygować tę niedogodność można użyć zakresów. Aby numery próbek zgadzały się z przyjętym w przetwarzaniu sygnałów zwyczajem, że początkowa próbka ma indeks 0, należy współrzędne poziome punktów określić jako liczby od 0 do 10.[2] Można to załatwić podając jako pierwsze dwa argumenty funkcji `plot` dwie tablice: jedną z wartościami współrzędnych poziomych (odciętych) a drugą na współrzędne pionowe (rzędne) punktów. Ta pierwsza może być podana w postaci zakresu `0:10` lub w postaci ogólnej:

```
plot(0:length(x)-1, x, "o");
```

gdzie pierwszy argument funkcji `plot` to zakres od 0 do liczby określonej długością tablicy `x` (w naszym wypadku jest to 11) pomniejszona o 1. Uzyska się w ten sposób wykres z rysunku 1b.

Dodatkowo, często chcemy wykreślić punkty tworzące sygnał dyskretny na wykresie, na którym pozioma oś to nie numery próbek a czas wyrażony w jednostkach czasu (np. sekundach), z którego próbki te pochodziły. Przyjmijmy, że częstotliwość próbkowania $f_s = 25\,\mathrm{Hz}$. Oznacza to, że próbka o numerze 0 pochodzi z czasu 0, próbka o numerze 1 pochodzi z czasu równego $\frac{1}{25}$s, próbka 2 z czasu równego $\frac{2}{25}$s, itd. Nieznacznie zatem modyfikując wyrażenie na wartości odciętych[3] można wykreślić te same dane ale w funkcji czasu, jak na rysunku 1c.

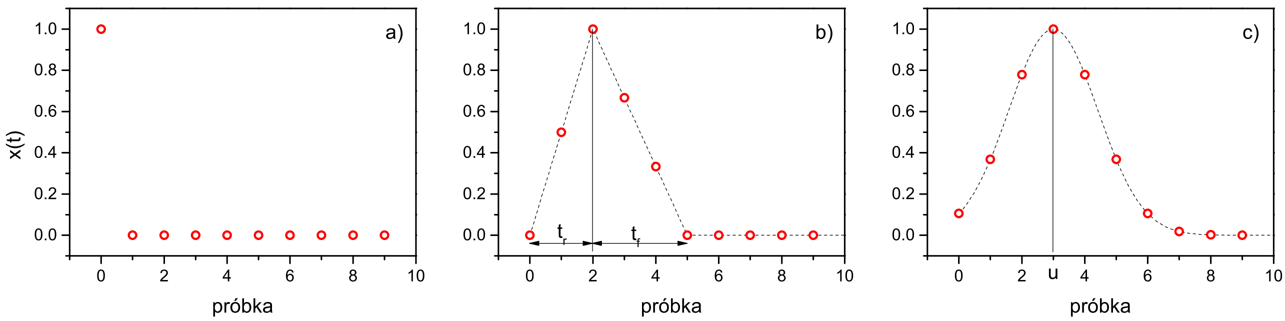
[1] Drugi parametr: "o" powoduje, że wykres składa się tylko z okrągłych i pustych w środku punktów. Istnieje wiele sposobów na formatowanie wykresów, na przykład "-o" da wykres połączonych liniami punktów, itd. Więcej szczegółów znajdziesz w pomocy do funkcji `plot`.

[2] Zwróć uwagę, że 11 próbek sygnału ma indeksy od 0 do 10 a nie do 11, to często popełniany błąd!

[3] Pozostawiamy to do wymyślenia Tobie.

1.2 Wybrane sygnały impulsowe

W cyfrowym przetwarzaniu sygnałów używa się pewnych typowych sygnałów, jak sygnał harmoniczny, stały itp. Niektóre z typowych sygnałów impulsowych przedstawiono na rysunku 2.



Rysunek 2: Wybrane typowe sygnały impulsowe: a) impuls Kroeneckera, b) impuls trójkątny $t_r = 2$, $t_f = 3$, c) impuls Gaussa $u = 3$, $s = 1$.

W impulsie Kroeneckera początkowa próbka sygnału równa jest 1, pozostałe próbki to 0.

Impuls trójkątny to sygnał, w którym wartości próbek najpierw liniowo narastają od 0 do wartości maksymalnej w czasie narostu t_r, a następnie opadają do 0 w czasie opadania t_f.

Impuls Gaussa wyrażony jest krzywą Gaussa, którą można przedstawić w dziedzinie czasu t lub numeru próbki n jako:

$$x(t) = e^{-\frac{(t-u)^2}{2s^2}} \tag{1}$$

$$x[n] = e^{-\frac{(n-u)^2}{2s^2}}, \tag{2}$$

gdzie u to położenie środka impulsu (jego czas bądź numer próbki) a s jest parametrem wyrażającym szerokość impulsu.

1.3 Splot

Splotem sygnałów nazywamy działanie, którego argumentami są dwa sygnały i które zdefiniowane jest dla sygnałów ciągłych następującym wzorem:

$$s(t) = x_1(t) * x_2(t) = \int_{-\infty}^{\infty} x_1(u)x_2(t-u)du \tag{3}$$

Dla sygnałów dyskretnych x i y operację wyznaczania n-tej próbki splotu zapisać można następująco:

$$z[n] = x[n] * y[n] = \sum_{k=0}^{N_y-1} y[k]x[n-k] \tag{4}$$

Długość splotu dwóch sygnałów dyskretnych x, y wynosi $N_x + N_y - 1$, gdzie N_x jest długością sygnału x a N_y długością sygnału y. Obliczenie splotu wymaga zatem zsumowania iloczynów odpowiednich próbek splatanych sygnałów. To, które mają być mnożone wynika z indeksów sygnałów w wyrażeniu sumy. Przy obliczaniu splotu należy zwrócić uwagę na to, że kiedy indeksy sygnałów x i y wychodzą poza dozwolony zakres, to wartości sygnału zastępuje się zerami. Ilustrację tego procesu przedstawiono w Tabeli 1.

Tabela 1: Ilustracja obliczenia elementu $n = 0$, $n = 2$ i $n = 5$ splotu sygnałów x i y o długościach, odpowiednio, $N_x = 3$, $N_y = 5$. Tabela zawiera wartości indeksów sygnałów k i $n - k$, symbole próbek sygnałów x i y o tych indeksach (lub 0 jeśli brak danej próbki) oraz niezerowe iloczyny $y[k]x[n - k]$, których suma da n-tą wartość splotu $x * y$.

$n = 0$									
k	-3	-2	-1	0	1	2	3	4	5
$y[k]$	0	0	0	y_0	y_1	y_2	y_3	y_4	0
$n - k$	3	2	1	0	-1	-2	-3	-4	-5
$x[n - k]$	0	x_2	x_1	x_0	0	0	0	0	0
$y[k]x[n - k]$	0	0	0	y_0x_0	0	0	0	0	0

$n = 2$									
k	-3	-2	-1	0	1	2	3	4	5
$y[k]$	0	0	0	y_0	y_1	y_2	y_3	y_4	0
$n - k$	5	4	3	2	1	0	-1	-2	-3
$x[n - k]$	0	0	0	x_2	x_1	x_0	0	0	0
$y[k]x[n - k]$	0	0	0	y_0x_2	y_1x_1	y_2x_0	0	0	0

$n = 5$									
k	-3	-2	-1	0	1	2	3	4	5
$y[k]$	0	0	0	y_0	y_1	y_2	y_3	y_4	0
$n - k$	8	7	6	5	4	3	2	1	0
$x[n - k]$	0	0	0	0	0	0	x_2	x_1	x_0
$y[k]x[n - k]$	0	0	0	0	0	0	y_3x_2	y_4x_1	0

1.4 Właściwości splotu

Działanie splotu, zarówno dla sygnałów ciągłych i dyskretnych, jest przemienne, czyli dla sygnałów x, y i z:

$$x * y = y * x \tag{5}$$

łączne, czyli:

$$x * (y * z) = (x * y) * z \tag{6}$$

rozdzielne względem dodawania, czyli:

$$x * (y + z) = x * y + x * z \tag{7}$$

oraz łączne względem mnożenia przez skalar c, czyli:

$$c \cdot (x * y) = (c \cdot x) * y = x * (c \cdot y) \tag{8}$$

1.5 Splot sygnału z deltą

Na wykładzie usłyszycie lub już usłyszeliście o tym, jaki efekt daje splot delty Diraca $\delta(t-t_0)$ z sygnałem ciągłym $y(t)$. Podobnie zachowuje się to działanie dla delty Kroneckera $\delta[n - n_0]$ i sygnału dyskretnego $y[n]$.

Oblicz sobie, na przykład korzystając ze sposobu z tabeli 1, jaki będzie efekt splotu $z = x*y$, jeśli $x[n] = \delta[n] = [1, 0, 0]$. Licząc wartość splotu dla $n = 0$ jedyna niezerowa próbka $x[n]$ będzie mnożona z $y[0]$. Dla $n = 1]$ będzie to iloczyn z $y[1]$, i tak dalej. W rezultacie uzyskamy sygnał $z[n] = y[n]$.

Jeśli $x[n]$ do delta Kroneckera opóźniona o jedną próbkę, czyli $x[n] = \delta[n - 1] = [0, 1, 0]$, to "jedynka" z $x[n]$ pomnożona będzie z początkową próbką sygnału y czyli z $y[0]$ dopiero gdy obliczać będziemy drugą próbkę splotu $z[1]$. Licząc $z[2]$ "jedynka" delty pomnożona będzie z

$y[1]$, i tak dalej. W rezultacie uzyskamy $z[n]$ będący opóźniony o jedną próbkę sygnał $y[n]$, czyli $z[n] = y[n-1]$.

Ogólnie, jeśli $z[n] = \delta[n-n_0] * y[n]$, czyli jeżeli obliczamy $z[n]$ będący splotem delty Kroneckera opóźnionej o n_0 próbek z sygnałem $y[n]$, to uzyskamy $z[n] = y[n-n_0]$, czyli sygnał y opóźniony o n_0 próbek.

Zastanów się, wracając do właściwości splotu, jaki będzie wynik splotu $z[n] = x[n] * y[n]$ jeżeli $x[n] = \delta[n-n_0] + \delta[n-n_1]$, czyli jeżeli jednym ze splatanych sygnałów jest suma dwóch delt Kroneckera, jedna opóźniona o n_0, a druga o n_1. A co, jeżeli $x[n]$ jest sumą jeszcze większej liczby delt Kroneckera każda opóźniona o inną liczbę próbek?

2 Funkcje do przygotowania przed zajęciami

Należy poprawić, przetestować oraz przygotować następujące funkcje narzędziowe: `gen_time`, `gen_sin`, `gen_delta`, `gen_triangle`, `gen_gauss` oraz `sig_delay_N`, w których kodzie znajdują się drobne błędy. Funkcje te i ich prawidłowe działanie potrzebne będą w kolejnych ćwiczeniach.

Proszę pamiętać o tym, że w jednym pliku ze skryptem może znajdować się jedna funkcja, oraz że nazwa pliku musi dokładnie odpowiadać nazwie funkcji (np. implementacja funkcji `gen_time` musi być zapisana w pliku `gen_time.m`, `gen_sin` w `gen_sin.m`, itd.).

Pośród funkcji narzędziowych znajdują się:

- `gen_time`, stworzona na podstawie informacji z punktu 1.1, która na podstawie podanej liczby próbek `N` i częstotliwości próbkowania `fs` generuje jednowymiarowy wektor, zawierający czasy kolejnych próbek, przy czym czas pierwszej z nich powinien wynosić `0`. Wartości tego wektora mają służyć jako wartości odciętych `N` próbek sygnału na ich wykresach w funkcji czasu

```
function time = gen_time(N, fs)
  time = zeros(1, N);
  for i=1:N
    time(i) = i/fs;
  endfor
endfunction
```

  W fazie pracy nad funkcją `gen_time` można na przykład wykreślić jej wynik w funkcji `I=1:N` korzystając z polecenia `plot(I, time)` by sprawdzić czy wektor `time` zaczyna się od zera, lub wydrukować na ekranie wektor `time` w command window lub też sprawdzić pierwszy element wektora `time` w workspace.

- `gen_sin`, która na podstawie wygenerowanego wektora z czasami próbek `time` generuje wektor z przebiegiem sinusoidalnym o częstotliwości `fsin`, amplitudzie `A` i przesunięciu fazowym `fi`

```
function signal=gen_sin(time, fsin, A, fi)
  signal = A*sin(time + fsin + fi);
endfunction
```

  Podczas pracy nad funckją sprawdź składnię funkcji wbudowanej `sin()`. Dla ustalonych parametrów np. `fsin=0.1, A=1, fi=0` wykreśl jeden okres `gen_sin()` używając np. polecenia `plot(time,signal)`.

- gen_delta, która na podstawie wygenerowanego wektora z czasami próbek time generuje deltę Kroeneckera

```
function signal = gen_delta(time)
  N = length(time);
  signal = zeros(N);
  signal(0) = 1;
endfunction
```

  Sprawdź definicję funkcji delty Kroneckera (Diraca), popraw funkcję gen_delta, a następnie sporządź jej wykres korzystając np. z funkcji stem(time,signal).

- gen_triangle, która na podstawie wygenerowanego wektora z czasami próbek time generuje impuls trójkątny o amplitudzie A, czasie narostu tr i czasie opadania tf

```
function signal = gen_triangle (time, A, tr, tf)
  N = length(time);
  signal = zeros(1, N);
  for n = 1:N
    if time(n) < tr # jezeli trwa narost
      if (tr<0)
        signal(n) = A*time(n)/tr;
      endif;
    elseif time(n) <= tr+tf # jezeli trwa spadek
      if (tf>0)
        signal(n) = A - A*(time(n)-tr)/tf;
      else/
        signal(n) = A;
      endif
    endif
  endfor
endfunction
```

  W fazie testów nad poprawnym działaniem funkcji dobierz tak parametry (czas narastania oraz opadania), by wykreślić np. trójkąt równoramienny.

- gen_gauss, która na podstawie wygenerowanego wektora z czasami próbek time generuje impuls Gaussa, którego środek przypada na czas u a szerokość opisano parametrem s

```
function signal=gen_gauss(time, u, s)
  N = length(time);
  signal = zeros(1,N);
  for n=1:N-1
    signal(n) = exp(-(time(n)-u)^2/2*s^2);
  endfor
endfunction
```

  prawdź definicję funkcji Gaussa (rozkładu normalnego). W trakcie pracy nad ciałem funkcji wykonaj kilka testów: np. wyśrodkuj krzywą względem zmiennej time, dla ustalonego parametru u definiującego średnią, wykonaj testy z parameterem s, określającym odchylenie standardowe (np. zmniejsz s dziesięciokrotnie względem pierwotnej wartości).

- `sig_delay_N`, która opóźnia sygnał `x` o `Nd` próbek

  ```
  function y = sig_delay_N(x, Nd)
    N = length(x);
    y = zeros(1, N);
    for i=0:N-Nd-1
      y(i+Nd) = x(i);
    endfor
  endfunction
  ```

  Po analizie i poprawie kodu w fazie testów nad poprawnym działaniem `sig_delay_N` dla ustalonego `Nd` wykreśl wynik działania dwóch funkcji: `gen_delta` oraz `gen_delta_N` korzystając np. z funkcji `stem(time,signal, time, y)`. Czy poprawiona funkcja działa prawidłowo?

Wskazówki do testowania funkcji narzędziowych

Funkcje najwygodniej jest testować za pomocą osobnego skryptu zapisanego w pliku. Jego nazwa nie ma znaczenia. Na przykład zawartość skryptu testującego funkcje z zadania 1. może wyglądać następująco (skrypt jest niedokończony, w miejsce wielokropków należy wpisać parametry generowanych sygnałów czy operacji na sygnałach):

```
time = gen_time(...); # generacja tablicy z czasami próbek
x1 = gen_sin(time, ...); # generacja przebiegu sinusoidalnego
x2 = gen_triangle(time, ...); # generacja impulsu trójkątnego
x3 = gen_delta(time); # generacja delty Kroeneckera
x3d = sig_delay_N(x3, ...); # opóźniona delta Kroeneckera

#wykreslenie wygenerowanych sygnalow
figure(1);
plot(time, x1, time, x2); # sinus i trójkąt na rysunku 1
figure(2);
plot(time, x3, time, x3d); # ilustracja przesunięcia delty Kroeneckera
```

Generując sygnały dostosuj czasy trwania bądź częstotliwości do częstotliwości próbkowania. Aby przebieg sygnału dyskretnego w funkcji czasu był czytelny kryterium Shannona - Nyquista musi być spełnione z dużym nadmiarem.

Przygotowania/testy dotyczące operacji splotu.

Ponadto po analizie podrozdziałów 1.3– 1.5 wyznacz analitycznie (ręcznie) splot `x*y` zgodnie z Tabelą 1 dla np. `x = [1,2,3], y=[1,2]`.

Zapoznaj się ze składnią funkcji wbudowanej `conv(x,y)` a następnie sprawdź poprawność swoich wyliczeń korzystając z funkcji wbudowanej.

Dla zadanych wektorów np. `x = [1,2,3], y=[1,2], z=[2,1]` sprawdź porawność własności splotu podanych w podrozdziale 1.4. Zastanów się czego konsekwencją są własności (5), (7) oraz (8)? Potrafisz znaleźć matematyczne uzasadnienie/wyjaśnienie tych wzorów np. w oparciu o rachunek całkowy?

Zdefiniuj wektory: δ`{n} = [1,0,0,0]` deletę Kroneckera oraz dowolny wektor na przykład `x = [1,2,3,4]`. Następnie wykonaj ręcznie operację splotu (korzystając np. z tablicy 1) oraz sparwdź jej wynik w oparciu o funkcję `conv()`. W jaki sposób wpływa funkcja delty Kreneckera na elementy wektora `x`?

3 Origin - poznawane funkcje

`sin(x)` - wyznaczanie funckji sinus
`cos(x)` - wyznaczanie funckji cosinus
`stem(x,y)` - wykreślanie widma sygnału
`plot(x,y)` - wykreślanie funkcji interpolowanej liniowo
`conv({x}, {y})` - wyznacza splot dwóch wektorów

`figfile_store()` - zapis danych w postaci wektora (2x1) lub (1x2) do pliku;
przykładowe wywołanie: `figfile_store('dane',x,y);`
tworzy plik o nazwie dane.dat zawierający x oraz y

`figfile_savepng()` - wczytanie danych z pliku oraz wegenerowanie wykresu w postaci `plot()` lub `stem()`; przykładowe wywołanie:
`figfile_savepng(0, 'dane', 'fig1', 't(s)', 'u(V)', 'napięcie')`
wykreśla wykres typu `plot()` z danych zapisanych w pliku dane.dat w postaci fig1.png

4 Pytania i zadania na kartkówkę

1. Ile próbek sygnału dyskretnego uzyska się próbkując z częstotliwością 3 kHz przez 300 ms?
2. Jaką częstotliwość próbkowania trzeba zastosować, aby w ciągu 7 sekund zebrać 56000 próbek?
3. Ile czasu upłynie od rozpoczęcia próbkowania do zebrania 16500 próbki, gdy częstotliwość próbkowania to 44100 Hz?
4. Dane są tablice: `a = [2 3 8 1 3 6 5 1 2 3 7]` i `b = [8 1 4 2 9 7 1 2 3 4 2]`. Oblicz ręcznie wartość $d = \sum_{i=0}^{4} a_{i+5} \cdot b_{8-i}$ przyjmując, że indeksy początkowych elementów w tablicach to a_0 i b_0.
5. Jak będzie wyglądał fragment skryptu Octave, który zrealizuje działanie na elementach jednowymiarowych tablic `a` i `b` dokładnie według wzoru: $d = \sum_{i=0}^{4} a_{i+5} \cdot b_{8-i}$. Zmienne w skrypcie mają mieć takie same nazwy jak we wzorze, a początkowe elementy w tablicach to a_0 i b_0.
6. Jaką długość będzie miał splot sygnałów o długości 10 i 30 próbek?
7. Jaką wartość będzie miała 4. (licząc od 0) próbka splotu sygnałów $x * y$, jeżeli `x=[1 5 4 3 2 3 4 5 6]` i `y=[1 3 0 1]`?
8. Jaką wartość będzie miała 3. (licząc od 0) próbka splotu sygnałów $x * y + x * z$, jeżeli `x=[1 5 4 3 2 3 4 5 6]`, `y=[1 3 0 1]` i `z=[2 5 1 2]`?
9. Jaki uzyskuje się wynik splotu, jeżeli jednym ze splatanych sygnałów jest jednak lub suma większej liczby delt Kroneckera opóźnionych każda o inną liczbę próbek?

Uwaga. Wartości liczbowe podane w pytaniach są przykładowe. Na kartkówce podobne zadania będą zawierały inne dane.

5 Zadania do realizacji na zajęciach

5.1 Funkcje narzędziowe

W trakcie zajęć sprawdzana będzie znajomość implementacji, prawidłowość działania i umiejętność korzystania z funkcji opisanych w poniższych zadaniach. Obowiązkiem jest przygotowanie ich przed przyjściem na zajęcia.

Każda z definicji wykonuje operacje, które zostały teoretycznie omówione w poprzednich rozdziałach niniejszej instrukcji. Za każde zadanie można otrzymać podaną liczbę punktów pod warunkiem, że zostanie ono w całości poprawnie zrealizowane.

Na zajęciach należy wygenerować i przeanalizować trzy typy sygnałów wskazanych przez prowadzącego: po dwa wykresy dla każdego typu sygnału zmieniając zestaw parametrów wejściowych. Na przykład: kilka okresów funkcji sin() o określonej amplitudzie oraz przesunięciu fazowym: `A=1..5 fi=`$0..2\pi$, wyśrodkowany wykres Gaussa czy wykres trójkątny dla ustalonych parametrów $u = 0.1..0.5$ oraz $s = 0.01...0.05$, oraz odpowiednio czasu narastania oraz opadania `tr` oraz `tf`, itd.

Wszystkie funkcje, których argumentem bądź wynikiem jest sygnał w postaci wektora, powinny w pierwszym jej elemencie (czyli elemencie o indeksie 1) zawierać pierwszy element sygnału (dla $t = 0$).

Wszystkie funkcje powinny być napisane czytelnie, z zachowaniem zasad formatowania kodów źródłowych języków wysokiego poziomu.

Prawidłowe działanie funkcji należy zilustrować za pomocą wykresów generowanych przebiegów w funkcji czasu. Za ich wykonanie otrzymuje się 2 pkt.

W sprawozdaniu powinny się znaleźć wygenerowane wykresy dla określonych zestawów parametrów wejściowych, a także komentarz dotyczący otrzymanych wyników, np. czy funkcję zachowują się zgodnie z oczekiwaniami przy zmianie parametrów wejściowych?

Odsłuchiwanie sygnałów okresowych

Dodatkowym testem, który mogą Państwo wykonać po wykonaniu tego ćwiczenia, mającym też charakter zabawy, jest odsłuchanie sygnałów tworzonych w Octave przez głośniki komputera. Aby móc to zrobić należy generować sygnały o częstotliwościach próbkowania idących w dziesiątki kHz, na przykład powszechną w cyfrowym audio częstotliwość $f_s = 44.1\,\mathrm{kHz}$. Dźwięk taki musi trwać sekundę czy kilka sekund, musi zatem zawierać odpowiednią do tego liczbę próbek.

Zakładając, że w wektorze `x` zapisano wartości z przedziału od -1 do 1 odpowiadające próbkom dźwięku (na przykład utworzonych funkcją `gen_sin(...)`), a zmienna `fs` równa jest wyrażonej w hercach częstotliwości próbkowania, można ten dźwięk odtworzyć za pomocą następujących funkcji Octave:

```
#utworzenie odtwarzacza audio probek sygnalu x z czestotliwoscia probkowania fs
player = audioplayer(x, fs);
#uruchomienie odtwarzacza
playblocking (player);
```

Uwaga, dźwięk wykorzystujący cały przedział wartości od -1 do 1 może być bardzo głośny. Ostrożnie zatem z ustawieniami głośności w komputerze, zwłaszcza przy korzystaniu ze słuchawek!

Metodę tę można zastosować do odsłuchiwania dźwięków o różnych częstotliwościach, na przykład odpowiadających poszczególnym dźwiękom, lub też współbrzmienion, czyli sumom poszczególnych dźwięków tworzących akordy. Więcej informacji na ten temat można znaleźć w Wikipedii: https://pl.wikipedia.org/wiki/D%C5%BAwi%C4%99k_(muzyka), https://pl.wikipedia.org/wiki/Tr%C3%B3jd%C5%BAwi%C4%99k.

5.2 Operacje arytmetyczne na sygnałach i parametry sygnałów

Wygeneruj wektor `time` zawierający czasy próbek z parametrami podanymi przez prowadzącego. Używając funkcji z poprzedniego punktu należy wygenerować kilka różnych sygnałów.

Zaprezentuj wskazane przez prowadzącego działania na sygnałach, takie jak na przykład:

- dodawanie sygnałów,
- mnożenie sygnałów,
- dodawanie składowej stałej do sygnału,
- zmianę amplitudy sygnału przez jego pomnożenie przez stałą.
- przesuwanie sygnału w czasie o wskazaną liczbę próbek, sumowanie sygnałów przesuniętych o różne opóźnienia,
- obliczenie ich wskazanych parametrów, jak na przykład: wartość średnią, wartość skuteczną czy inne.

Prawidłowe wykonanie operacji arytmetycznych należy zilustrować za pomocą wykresów. Najwygodniej stworzyć skrypt, który będzie te operacje arytmetyczne wykonywał oraz wyświetlał wykresy. Wykonanie zadania otrzymuje się 1 pkt.

5.3 Splot

Wykorzystując znojomość własności splotu, spośród dostępnych funkcji narzędziowych generujących sygnały, zaproponuj wybór przynajmniej jednej pary sygnałów, których wynikowy splot świadczy o prawidłowym wykonaniu operacji splotu. Do realizacji zadania użyj funkcji wbudowanej `conv(x,y)`. Działanie zilustruj wykonując operację splotu na dwóch sygnałach, których splot jest łatwy do obliczenia bez użycia komputera. Co to za sygnały? Przypomnij sobie informacje o splocie przedstawione na wykładzie.

Zademonstruj prawdziwość własności wzoru (5) lub (7) podanych w Sek. 1.4 i zastanów się z czego ona wynika? Spróbuj wyjaśnić matematycznie prawdziwośc tych własności.

[4]*Przanalizuj poniższą funkcję i usuń błędy z funkcji `sig_conv` obliczającej wartość splotu dwóch sygnałów `x` i `y`. Zwróć uwagę na wartości wyniku oraz na długość sygnału wynikowego. Ponownie wybierz takie dwa sygnały, których splot jest łatwy do obliczenia analitycznie.

Wyniki przedstaw na wykresach. Za prawidłowe wykonanie zadania otrzymasz 2 pkt.

```
function z = sig_conv(x,y)
  Nx = length(x);
  Ny = length(y);
  z = zeros(1,Nx+Ny); #utworzenie pustej tablicy na wynik splotu
  for n=0:Nx+Ny-1 # dla każdego elementu tablicy wynikowej
    c=0; # zsumuj odpowiednie iloczyny probek sygnalow
    for k=0:Ny-1
      if ((n-k)<=0) && ((n-k)<Nx) # jesli pozwalaja na to indeksy probek
        c = c + x(n-k)*y(k);  # dodaj do c odpowiedni iloczyn
      endif
    endfor;
    z(n+1) = c; # zapis wyniku w tablicy
  endfor;
endfunction
```

[4]Zadanie dodatkowe o podwyższonym stopniu trudności.