Przetwarzanie sygnałów

Ćwiczenie 1

Wprowadzenie do programu Octave

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

Spis treści

Katedra Nanometrologii
Wydział Elektroniki, Fotoniki i Mikrosystemów
Politechnika Wrocławska

1 Wprowadzenie do programu Octave

Mimo że program Octave został stworzony do obliczeń na macierzach, na zajęciach będzie można korzystać z operacji na macierzach tylko w przypadku importu/eksportu danych i w niektórych zadaniach projektowych, wskazanych specjalnie w instrukcji. W przypadku implementacji własnych algorytmów przetwarzania (np. DFT, FFT) wolno korzystać z macierzy tak jak ze zwykłych tablic w języku C. Dlatego wszystkie definicje operatorów są podane dla wielkości skalarnych. Odpowiednie operatory macierzowe można znaleźć w anglojęzycznej dokumentacji.

1.1 Operatory

Operatory arytmetyczne

`x+y` operator sumy
`x-y` operator różnicy
`x*y` operator mnożenia
`x/y` operator dzielenia
`x.*y` operator mnożenia elementów tablic
`x./y` operator dzielenia elementów tablic
`-y` zmiana znaku na przeciwny
`a^b` podniesienie do potęgi
`x++` inkrementacja (to samo co `x=x+1`)
`x--` dekrementacja

Operatory logiczne

`x&y` iloczyn logiczny
`x|y` suma logiczna
`~x` negacja logiczna
`x||y` suma logiczna (jeśli x jest prawdziwe, to y nie jest obliczane)
`x&&y` iloczyn logiczny (jeśli x jest 0, to y nie jest obliczane)
`!x` negacja logiczna
`~x` negacja logiczna

Operatory relacji

`x<y` operator mniejszości
`x<=y` operator mniejsze równe
`x==y` operator równości
`x>y` operator większości
`x>=y` operator większe równe
`x!=y`, `x<>y`, `x~=y` operatory nierówności

Operatory przypisania

`x=y` przypisanie wartości y do x
`x+=y` przypisanie z dodawaniem (to samo co x=x+y)
`x-=y` przypisanie z odejmowaniem
`x*=y` przypisanie z mnożeniem
`x/=y` przypisanie z dzieleniem

Operatory wykonywane są zgodnie z odpowiednimi regułami pierwszeństwa. Można wymusić odpowiednią kolejność wykonania działań poprzez zastosowanie nawiasów.

1.2 Często używane funkcje

W obliczeniach używa się funkcji dostarczanych przez środowisko Octave. Poniżej wypisano niektóre z najczęściej przydających się w trakcie realizacji ćwiczeń laboratoryjnych:
`sin (x)` obliczenie sinusa (argument w radianach)
`cos (x)` obliczenie sinusa (argument w radianach)
`abs (x)` obliczenie wartości bezwzględnej liczby rzeczywistej bądź modułu liczby zespolonej
`arg (x)` obliczenie argumentu liczby zespolonej (wyrażonego w radianach)
`real (x)` obliczenie części rzeczywistej liczby zespolonej
`imag (x)` obliczenie części urojonej liczby zespolonej.

Jeżeli argumentami funkcji są tablice operacja operacja zostanie wykonana dla wszystkich elementów tablicy.

1.3 Tablice

Tablice tworzymy przypisując pod odpowiednie indeksy odpowiednie wartości. Octave sam dostosowuje rozmiary do użytych indeksów. `x(10)=0` utworzy tablicę `x` i pod indeks 10 wpisze zero;

Można też skorzystać z funkcji `zeros()`, służącą do utworzenia tablicy wypełnionej zerami. `x=zeros(1,10)` tworzy tablicę jednowymiarową o rozmiarze `10`, wypełnioną zerami.

Tablice można również tworzyć używając tzw. zakresów podając je w formie `b:i:e` lub `b:e`, gdzie `b` to wartość początkowa, `e` wartość końcowa a `i` różnica pomiędzy kolejnymi wartościami (jej niepodanie oznacza 1). Na przykład `x=1:2:10` utworzy tablicę wypełnioną liczbami `1, 3, 5, 7, 9` (od 1, zwiększane o 2, nie większe niż 10) a `x=1:5` utworzy tablicę wypełnioną liczbami `1, 2, 3, 4, 5`.

Tablice wielowymiarowe tworzymy i korzystamy z nich, podając odpowiednie indeksy po przecinku np. `x(1,10)`, odwołuje się pierwszego wiersza i dziesiątej kolumny tablicy dwuwymiarowej.

Tablice można dodawać i odejmować. Powoduje to utworzenie tablicy, w której elementy będą sumami bądź różnicami odpowiadających sobie elementów tablic źródłowych. Mnożenie tablic wykonywane za pomocą operatora mnożenia dokonuje mnożenia macierzowego. Jeżeli chcemy pomnożyć odpowiadające sobie elementy dwóch tablic `a` i `b` należy użyć operatora `a .* b`.

1.4 Instrukcje sterujące wykonaniem programu

Instrukcja if

```
if (x<y)
  x=10;
  y=100;
endif
```

```
if (x<y)
  x=10;
  y=100;
else
  x=11;
  y=111;
endif
```

```
if (x<y)
  x=10;
  y=100;
elseif (x==y)
  x=0;
  y=0;
else
  x=11;
  y=111;
endif
```

Instrukcja switch

```
switch x
  case 1
    y(1)=1;
    y(2)=2;
  case 2
    y(1)=3;
    y(2)=4;
  otherwise
    y(1)=0;
    y(2)=0;
endswitch
```

Jeśli x będzie równe 1, to y będzie tablicą dwuelementową zawierającą 1 i 2. Gdy x będzie równe 2, to w y będzie 3 i 4 itd.

Instrukcje `if` i `switch` mają takie samo znaczenie jak w językach C lub w C++. Istnieje drobna różnica w przypadku instrukcji `switch`, ponieważ wymienione przypadki wykluczają się. W języku C aby zapewnić wykluczanie wykorzystywało się instrukcję `break`. W środowisku Octave do obsługi wielu przypadków tym samym zestawem instrukcji używa się nawiasów klamrowych:

```
switch x
  case {1,2,4}
    %ta część wykona się, gdy x jest równe 1, 2 albo 4
  otherwise
    %ta część wykona się w pozostałych przypadkach
endswitch
```

Pętle programowe

```
while (x<y)
  x+=10;
  y-=10;
endwhile
```

```
do
  x+=10;
  y-=10;
until (x>y)
```

```
for x=y
  p(x)=0;
endfor
```

Pętla `while` wykonuje się, dopóki wyrażenie w nawiasie jest prawdziwe.

Pętla `do until` wykonuje się, dopóki wyrażenie w nawiasie jest fałszywe (podobnie jak w języku Pascal)

Pętla `for` w środowisku Octave jest ściśle związana operacjami na macierzach. Jeśli `y` jest wektorem wierszowym, to pętla for wykona się tyle razy, ile elementów ma wektor `y`. W każdej iteracji za `x` podstawiane będą kolejne elementy wektora `y`. Jeżeli `y` będzie wektorem kolumnowym lub macierzą (nie ważne ile wymiarową), to w każdej iteracji `x` będzie kolejnym wektorem kolumnowym macierzy `y`, a pętla wykona się tyle razy, ile takich wektorów kolumnowych zawiera macierz `y`. Najczęściej jednak pętlę for wywołuje się wykorzystując wektory wierszowe utworzone za pomocą omówionych wcześniej zakresów:

```
for i=1:10
  x(i)=0;
endfor
```

```
for i=1:2:10
  x(i)=2;
endfor
```

```
for i=10:-1:1
  x(i)=0;
endfor
```

Pierwsza pętla ta wykonuje się 10 razy i w każdej iteracji `i` przyjmuje kolejne wartości od 1 do 10. W drugiej krok iteracji zwiększono do 2. Trzecia pętla odliczana jest wstecz: od wartości 10 wartość `i` jest zmniejszana o 1 aż do 1.

1.5 Wykresy

Dane źródłowe do wykresów muszą być zawarte w tablicach. Jeśli `y` jest wektorem wierszowym, zawierającym kolejne wartości do przedstawienia na wykresie, to instrukcja `plot(y)` spowoduje przedstawienie jej na wykresie, na którym współrzędne `x` stanowią indeksy w wektorze.

Do wykonania wykresu funkcji potrzebne są dwa wektory wierszowe. Np mając wektor wierszowy `x`, zawierający kolejne wartości zmiennej niezależnej funkcji, oraz `y`, zawierający wartości funkcji rozkaz `plot(x,y)` spowoduje przedstawienie funkcji na wykresie.

Chcąc przedstawić większą liczbę wykresów na tym samym układzie współrzędnych można użyć instrukcji `plot(x1, y1, x2, y2)`. Jej wykonanie spowoduje wykreślenie wykresów dwóch

funkcji, których wartości zmiennej niezależnej i wartości funkcji znajdowały się w `x1` i `y1` dla jednej i `x2` i `y2` dla drugiej funkcji.

Instrukcja `figure` umożliwia stworzenie większej liczby wykresów. Np. po wykonaniu instrukcji `figure(1)` wszystkie kolejne instrukcje dotyczące m. in. wykreślania funkcji dotyczyć będą rysunku `Figure 1`, po `figure(2)` rysunku `Figure 2` itd.

1.6 Funkcje i skrypty

Zarówno funkcje, jak i skrypty są plikami, w których można zapisać programy w języku Octave. Skrypty służą przede wszystkim do zapisania sekwencji operacji w pliku, który można następnie uruchomić jednym poleceniem, będącym nazwą pliku.

Przykład zawartości skryptu zapisanego w pliku o nazwie `test.m`:

```
x=zeros(1,10);
for i=1:10
  x(i)=i;
endfor
```

Po wpisaniu w konsoli środowiska Octave polecenia `test` wykonany zostanie powyższy program. Funkcje to specjalne skrypty, które mogą pobierać argumenty i zwracać wartości. Definiowane są podobnie jak w środowisku Matlab, z tym że na końcu każdej funkcji dodaje się słowo kluczowe `endfunction`.

```
function [out1, out2, out3]=nazwa_funkcji(arg1,arg2,arg3)
#
# Tu definiuje się ciało funkcji
#
# tak przekazuje się odpowiednie zwracane wartości
  out1=10;
  out2=20;
  out3=30;
# wykonanie funkcji można zakończyć wcześniej za pomocą polecenia
# return
# należy wtedy pamiętać o uprzednim wpisaniu zwracanych wartości
# out1, out2, out3
# w przeciwnym wypadku wartości te nie będą ustalone
endfunction
```

Aby napisane skrypty czy funkcje były widoczne w środowisku, należy dodać zawierający je katalog do listy ścieżek pakietu Octave. Służy do tego polecenie `addpath` np.:

```
addpath("d:\\katalog1\\katalog2\\")
```

Jeżeli chcemy, aby ścieżki zostały dodane na stałe, to możemy je zapisać poleceniem `savepath`.

1.7 Operacje na plikach

Pliki podobnie jak w języku C, należy przed użyciem otworzyć za pomocą funkcji `fopen`:
`id=fopen("nazwa.txt", "r");`
a po użyciu zamknąć funkcją `fclose`:
`fclose(id)`

Funkcja `fopen` tworzy identyfikator pliku będący liczbą całkowitą. Zapis i odczyt z pliku odbywa się za pomocą tego identyfikatora i odpowiednich funkcji odczytu i zapisu. Najczęściej wykorzystywane będą funkcje `fprintf` i `fscanf`, które działają podobnie jak w języku C. Różnica polega na tym, że funkcje te zapisują i odczytują dane z macierzy, a nie z prostych wartości skalarnych np.:

```
x=[1.0 2.0 3.0 4.0 5.0 6.0];
id= fopen("plik.txt","w");
fprintf(id,"\%f",x);
fclose(id);
```

W pliku o nazwie plik.txt zostanie zapisany cały wektor `x`, przy czym liczby od 1.0 do 6.0 zostaną zapisane jedna po drugiej bez żadnego odstępu. Poniżej pokazano, jak ten odstęp dodać:

```
fprintf(id,"\%f ",x);
```

Po %f jest jeszcze jedna spacja, która jest dodawana po każdym elemencie wektora.

Jeżeli zmienna `x` jest macierzą, np.:

```
x=[1.0  2.0
3.0  4.0];
```

to funkcja

```
fprintf(id,"\%f ",x)
```

zapisze dane w następujący sposób 1.0 3.0 2.0 4.0, czyli zapisze najpierw pierwszą kolumnę, potem następną itd.

Kolejny przykład ilustruje, jak zapisać do pliku tekstowego wierszowe.txt trzy wektory wierszowe, a do pliku kolumnowe.txt – trzy wektory kolumnowe.

```
idw=fopen("wierszowe.txt",w);
idk=fopen("kolumnowe.txt",w);
w1=[1.0 2.0 3.0];
w2=[4.0 5.0 6.0];
w3=[7.0 8.0 9.0];
k1=[1.0
2.0
3.0];
k2=[4.0
5.0
6.0];
k3=[7.0
8.0
9.0];
fprintf(idw,"\%f \%f \%f\n", [w1;w2;w3]);
fprintf(idk,"\%f \%f \%f\n", [k1';k2';k3']);
fclose(idw);
fclose(idk);
```

Instrukcja `fscanf` zapisuje wczytane dane do wektora kolumnowego. Przykładowo: plik o nazwie `plik.txt` zawiera:

```
1.1 1.2 1.3
2.1 2.2 2.3
3.1 3.2 3.3
4.1 4.2 4.3
```

Jeżeli wykonamy dla tego pliku instrukcję: `x=fscanf(id,"%f")` to zawartość `x` będzie następująca:

```
x=[1.1
1.2
1.3
2.1
2.2
2.3
3.1
3.2
3.3
4.1
4.2
4.3];
```

Jeżeli chcemy, aby kolumny w pliku stały się kolumnami w macierzy, to instrukcja powinna wyglądać w następujący sposób:

```
x=fscanf(id,"\%f",[3 inf])';
```

Jeżeli chcemy teraz przepisać odpowiednie kolumny do odpowiednich wektorów kolumnowych, to używamy operacji indeksacji podobnie jak w środowisku Matlab:

```
k1=x(:,1); k2=x(:,2); k3=x(:,3)
```

2 Proste zadania programistyczne

Swoje umiejętności programowania możesz odświeżyć a specyfikę programu Octave przyswoić, pisząc nieskomplikowane skrypty realizujące następujące zadania:

1. W skrypcie umieść linię `x = [1 4 3 2 5];` czy podobną, tworzącą tablicę `x` z podaną zawartością. Napisz skrypt używający pętli `for` do obliczenia średniej z tej tablicy. Poszukaj potem w dokumentacji Octave funkcji wbudowanych w pakiet, które ułatwią to zadanie sprowadzając je do jednej linijki kodu.
2. Używając zakresów naucz się generować tablice zawierające określoną liczbę elementów `N` bądź pokrywających podany zakres wartości, przy czym kolejne elementy mają różnić się o `delta`. Na przykład zakres 15 elementów zaczynający się od 5, a różnica pomiędzy kolejnymi to 0,25 albo zakres od -4 do +3 elementów różniących się o 0,5.
3. Napisz skrypty rysujące wykresy funkcji kwadratowej czy wielomianu 3. stopnia.

3 Warunki zaliczenia laboratorium PS

Na ćwiczeniach laboratoryjnych oceniane jest: przygotowanie do ćwiczenia na podstawie kartkówki oraz wykonanie ćwiczenia na podstawie zaprezentowania wykonania zadań.

Ocena z kartkówki Na kartkówce student otrzymuje dwa pytania, za które uzyskuje maksymalnie po 2 pkt. W każdej instrukcji znajduje się lista pytań do danego ćwiczenia. Przeliczanie punktów na ocenę odbywa się w sposób następujący.

2 pkt	3,0
2,5 pkt	3,5
3 pkt	4,0
3,5 pkt	4,5
4 pkt	5,0

Ocena wykonania ćwiczenia Student w ramach pracy na laboratorium powinien wykonać 5 zadań. W przypadku każdego z nich powinien wykazać poprawność uzyskanych wyników i przejść pozytywnie weryfikację samodzielności pracy. Za wykonanie każdego zadania otrzymuje maksymalnie 1 pkt (razem maksymalnie 5 pkt). Przeliczanie punktów na ocenę odbywa się w sposób następujący.

3 pkt	3,0
3,5 pkt	3,5
4 pkt	4,0
4,5 pkt	4,5
5 pkt	5,0

Ćwiczenie jest zaliczone jeżeli student uzyska pozytywną ocenę zarówno z kartkówki jak i wykonania ćwiczenia. Wymagane jest zaliczenie wszystkich ćwiczeń. Ocena końcowa jest obliczana jako średnia z: średniej oceny z kartkówek oraz średniej oceny z ćwiczeń. Wartość ta zaokrąglana jest do najbliższej prawidłowej oceny. Kartkówki razem z protokołem wykonania ćwiczenia są jedynymi dokumentami umożliwiającymi skuteczną reklamację ocen.