LABORATORIUM OPROGRAMOWANIA UŻYTKOWEGO

MATLAB

Opracowanie:
dr inż. Jacek Kucharski
dr inż. Piotr Urbanek

ĆWICZENIE 9-10

Programowanie w MATLABIE

ELEMENTY JĘZYKA PROGRAMOWANIA

MATLAB jest językiem programowania o składni zapożyczonej z języka C. Zawiera on instrukcje warunkowe, instrukcje iteracyjne oraz instrukcje przerwania wykonywanych instrukcji. Dodatkowo funkcja error pozwala na tworzenie własnej diagnostyki błędów. Pełne zestawienie elementów języka można uzyskać poprzez polecenie help lang.

INSTRUKCJE

- **warunkowe**

```
if wyrażenie
  polecenia
elseif wyrażenie
  polecenia
else
  polecenia
end
```

Polecenia występujące po **if, elseif** lub **else** są wykonywane, jeśli macierz będąca wynikiem *wyrażenia* jest prawdą logiczną, tzn. jej wszystkie elementy są niezerowe. Części **elseif** oraz **else** są opcjonalne.

```
switch wyrażenie
        case wart_1
                polecenia
        case wart_2
                polecenia

        otherwise
                polecenia
end
```

Instrukcja porównuje wartość *wyrażenia* z wartościami występującymi po kolejnych słowach **case,** tj. *wart_1*, *wart_2*, itd., i w przypadku równości wykonywane są odpowiednie polecenia, przy czym dalsze przypadki występujące w instrukcji nie są już sprawdzane. Gdy w żadnym przypadku równość nie wystąpiła wykonywana jest cześć instrukcji po słowie **otherwise**, przy czym jest to część opcjonalna.

- **iteracyjne**

    1. z nieokreśloną liczbą obiegów pętli

    while *wyrażenie*
      polecenia
    end

Wykonuje polecenia tak długo jak długo wartość *wyrażenia* jest prawdą.

    2. z określoną liczbą obiegów pętli

    for *zmienna_iterowana=macierz_wartości*
      polecenia
    end

Wykonuje polecenia dla kolejnych wartości *zmiennej iterowanej*. Wartościami tymi są kolejne wektory kolumnowe pobrane z *macierzy wartości.* Jeżeli *macierz wartości* jest wektorem wierszowym to polecenia wykonywane są dla kolejnych elementów tego wektora. O uszeregowaniu wartości decyduje programista umieszczając je w odpowiedniej kolejności w *macierzy wartości*. Nie musza to być ciągi rosnące lub malejące.

- **instrukcja continue**

Powoduje przejście do wykonania kolejnej iteracji pętli (**for** lub **while**) z pominięciem, następujących po słowie **continue** poleceń. W przypadku pętli wielopoziomowych dotyczy to pętli, w której bezpośrednio się znajduje słowo **continue.**

- **instrukcja break**

Powoduje przerwanie wykonywania pętli, przy czym opuszczany jest tylko jeden poziom zagłębienia pętli.

- **instrukcja return**

Powoduje bezwarunkowe opuszczenie danej funkcji lub skryptu i powrót do miejsca jej wywołania.

M-PLIKI

Programy napisane w języku MATLABa umieszczane są w plikach tekstowych z rozszerzeniem .m (tzw. m-pliki) i mogą być tworzone przy użyciu dowolnego edytora tekstowego. Mogą one zawierać oprócz sekwencji poleceń i instrukcji MATLABa, wywołania innych m-plików, jak również wywołania samych siebie. Rozróżnia się dwa rodzaje m-plików: skryptowe i funkcyjne.

M-pliki skryptowe zawierają ciągi poleceń operujących na zmiennych globalnych zawartych w przestrzeni roboczej. Wykorzystuje się je w celu grupowania poleceń użytkownika ułatwiając ich wykonywanie np. w przypadku powtarzających się operacji. Skrypt nie posiada mechanizmu hermetyzacji tzn. dane utworzone w skrypcie zostaną dołączone do danych w przestrzeni roboczej i pozostają tam po zakończeniu wykonywania m-pliku skryptowego.
Komentarze w skrypcie poprzedza się znakiem %. Początkowy blok komentarzy stanowi treść pomocy dla danego skryptu po uzyskiwanej poleceniem help *nazwa_skryptu*.

M-pliki funkcyjne są to funkcje tworzone przez użytkownika operujące na zmiennych lokalnych i komunikujące się z przestrzenią roboczą poprzez parametry formalne. Takie m-pliki muszą się zaczynać od słowa kluczowego function. Pierwsza linia m-pliku funkcyjnego powinna być zapisana następująco:

function [*lista argumentów wyjściowych*]=nazwa_funkcji (*lista argumentów wejściowych*)

Nazwa funkcji musi być taka sama jak nazwa m-pliku, w którym się znajduje. Argumenty oddziela się przecinkami. Funkcja może nie mieć żadnych argumentów, wtedy nawiasy okrągłe obejmują listę pustą.
Argumenty wejściowe i wyjściowe oraz wszystkie zmienne używane wewnątrz funkcji mają charakter lokalny i nie są powiązane z przestrzenią roboczą (nawet jeśli mają takie same nazwy). Zmienne te są usuwane po zakończeniu wykonywania funkcji. Podobnie nie są „widziane" wewnątrz funkcji zmienne utworzone poza nią. Argumenty funkcji są przekazywane przez wartość, tzn. w ciele funkcji tworzona jest kopia zmiennych. Aby zwrócić wartość funkcji konieczne jest dokonanie odpowiednich przypisań wewnątrz funkcji.

Dwie funkcje standardowe zwracają liczby parametrów przekazywanych w aktualnym wywołaniu funkcji: nargin -wejściowych, nargout - wyjściowych. Można więc wywoływać funkcje z mniejszą niż zadeklarowana liczbą parametrów, ale trzeba określić co w takim przypadku ma zrobić funkcja.

Można także tworzyć funkcje o zmiennej, z góry nie określonej liczbie parametrów. W tym celu wykorzystuje się tablice komórkowe o nazwach *varargin i varargout,* w których zawarte są odpowiednie grupy argumentów.

```
function testvar(varargin)
for k = 1:length(varargin)
x(k) = varargin{k}(1); % Cell array indexing
y(k) = varargin{k}(2);
end
xmin = min(0,min(x));
ymin = min(0,min(y));
axis([xmin fix(max(x))+3 ymin fix(max(y))+3])
plot(x,y)

function [varargout] = testvar2(arrayin)
for k = 1:nargout
varargout{k} = arrayin(k,:) % Cell array assignment
end
```

M-plik może zawierać więcej niż jedną funkcję, przy czym pierwsza z nich pełni rolę funkcji podstawowej i jest wykonywana, gdy wywołana jest nazwa m-pliku, a pozostałe są funkcjami

lokalnymi, „widzianymi' jedynie wewnątrz danego m-pliku. Definicja każdej z funkcji zachowuje opisaną wcześniej strukturę.

Komentarze w funkcjach są wprowadzane podobnie jak w skryptach, a komentarz umieszczony bezpośrednio pod nagłówkiem funkcji będzie używany jako pomoc.

WYBRANE FUNKCJE STANDARDOWE

Funkcje do pracy interaktywnej:

input - wprowadzanie danych
menu - generowanie okna wyboru
pause - wstrzymanie wykonywania pliku

Funkcje do pomiaru czasu:

date - zwraca łańcuch zawierający aktualną datę,
clock - zwraca wektor postaci [rok miesiąc dzień godz min sek]
cputime - wyznacza czas procesora wykorzystany przez MATLABa od chwili uruchomienia.

```
tstart=cputime
   operacje
toper=cputime-tstart
```

etime(T1,T2) - zwraca różnicę czasu między parametrami wywołania, przy czym są one wektorem wyjściowym funkcji clock

```
tic
  operacje
toc
```

mierzy czas wykonania operacji i wyświetla go na ekranie.

Jeżeli zachodzi potrzeba obliczania funkcji zapisanych w postaci ogólnej to korzystamy z funkcji feval

feval(*nazwa_funkcji, x1,x2...xn*)

oblicza ona wartość funkcji o nazwie określonej łańcuchem znakowym dla podanych argumentów, np. feval('cos',[0:0.01:2pi])
Podobną funkcją jest eval, która pozwala na wykonanie poleceń MATLABa zapisanych w postaci łańcucha znaków.

Program ćwiczenia

1. Napisać m-pliki funkcyjne realizujące za pomocą wielkości skalarnych (iteracyjnie) wybrane jedno- i dwuargumentowe operacje macierzowe i tablicowe (+, *, .*, ′, .′ , ^, .^). Funkcje powinny sprawdzać rozmiary argumentów i informować o ewentualnych nieprawidłowościach. Należy także uwzględniać możliwość występowania skalarów.

2. Zbudować m-plik skryptowy będący nadrzędnym programem dla stworzonych w pkt. 1 m-plików funkcyjnych. Skrypt powinien umożliwiać:
   - wprowadzanie danych (argumentów) w wierszu poleceń – np. funkcja input,
   - wybór wykonywanej operacji – np. funkcja menu,
   - sprawdzenie poprawność wykonywanych przez m-pliki funkcyjne operacji wykorzystując wbudowane operatory macierzowe i tablicowe,
   - porównanie czasochłonności operacji realizowanej za pomocą m-pliku i operatora wbudowanego.

3. Napisać dwa m-pliki funkcyjne ze zmienną liczbą argumentów wejściowych i wyjściowych:
   - plik obliczający sumę lub iloczyn dowolnej liczby argumentów, przy czym jako pierwszy parametr wejściowy należy uwzględnić możliwość podawania (w postaci odpowiedniego symbolu) rodzaju wymaganej operacji (+, *, .*); program powinien sprawdzać rozmiary kolejnych argumentów, odrzucając te, które nie spełniają odpowiednich wymagań,
   - plik wykonujący transpozycję nieokreślonej z góry liczby macierzy, przy czym w przypadku argumentów zespolonych należy dla każdego z takich argumentów poprosić użytkownika o podanie rodzaju transpozycji.

4. Zbudować m-plik funkcyjny wyznaczający iloczyn macierzowy dwóch macierzy wielomianowych *A(s)* i *B(s)*. Do zapisu macierzy wielomianów należy wykorzystać tablice komórkowe. Należy ponadto wydzielić część programu realizującą mnożenie dwóch wielomianów i jako funkcję pomocniczą umieścić razem z funkcją podstawową w jednym m-pliku.

   Wskazówki:

   1. Odpowiednie tablice komórkowe powinny zawierać wektory, których elementami są współczynniki wielomianów uporządkowane według malejących potęg zmiennej *s*. Przykładowo, dla następujących macierzy *A*(s) i *B*(s):

$$A(s)=\begin{bmatrix} s^2+s & -2s^2+s+1 \\ -s^2+2s-1 & 2s^2+2 \end{bmatrix} \qquad B(s)=\begin{bmatrix} 2s^2+2 & s+3 \\ s-1 & \frac{1}{2}s+1 \end{bmatrix}$$

odpowiednie tablice komórkowe maja postać:

$$A = \begin{Bmatrix} [1,\ 1,\ 0] & [-2,\ 1,\ 1] \\ [-1,\ 2,\ -1] & [2,\ 0,\ 2] \end{Bmatrix} \qquad B = \begin{Bmatrix} [2,\ 0,\ 2] & [1,\ 3] \\ [1,\ -1] & [0.5,\ 1] \end{Bmatrix}$$

2. Obliczanie iloczynów dwóch wielomianów $a(s)$ i $b(s)$, na podstawie odpowiednich wektorów współczynników α, β zawartych w tablicach komórkowych A i B należy przeprowadzić według następującego algorytmu:

$$\gamma(k) = \sum_j \alpha(j)\beta(k+1-j)$$

gdzie:

γ - wektor współczynników wielomianu wynikowego,
j=max(1,k+1-n): min(k,m),
m – rozmiar wektora α,
n – rozmiar wektora β,

Przykładowo, dla wielomianów postaci:

$$a(s) = s^2 + 2s + 3; \quad b(s) = 4s^2 + 5s + 6$$

reprezentowanych następującymi wektorami:

$\alpha = [1, 2, 3]$; $\beta = [4, 5, 6]$;

obliczenia przebiegają następująco:

k=1
j = max(1,1+1-3): min(1,3) = 1:1=1
$\gamma(1) = \alpha(1) * \beta(1) = 1*4 = 4$;

k=2
j = max(1,2+1-3): min(2,3) = 1:2
$\gamma(2) = \alpha(1) * \beta(2) + \alpha(2) * b(1) = 1*5+2*4 = 5+8 = 13$;

k=3
j = max(1,3+1-3): min(3,3) = 1:3
$\gamma(3) = \alpha(1) * \beta(3) + \alpha(2) * \beta(2) + \alpha(3) * \beta(1) = 1*6 + 2*5 + 3*4 = 6 + 10 + 12 = 28$;

k=4
j = max(1,4+1-3): min(4,3) = 2:3
$\gamma(4) = \alpha(2) * \beta(3) + \alpha(3) * \beta(2) = 2*6 + 3*5 = 12+15 = 27$;

k=5
j = max(1,5+1-3): min(5,3) = 3: 3
$\gamma(5) = \alpha(3) * \beta(3) = 3*6 = 18$;

A zatem wektor wynikowy ma postać: γ=[4 13 28 27 18], co odpowiada wielomianowi: $c(s) = 4s^4 + 13s^3 + 28s^2 + 27s + 18$. Należy zauważyć, że rozmiar wektora γ wynosi (m+n-1),oraz słuszna jest zależność A(m x n)*B(n x p)=C(m x p).