Προγραμματισμός στο Matlab

Κλάδοι με την εντολή if

Σαν ένα απλό παράδειγμα κλάδου με την εντολή if ας θεωρήσουμε το παρακάτω παράδειγμα που υπολογίζει την απόλυτη τιμή ενός πραγματικού αριθμού.

```
function y = absval(x)
if x >= 0
   y = x;
else
   y = -x;
end
```

Εναλλακτικά μπορούμε να ορίσουμε

```
function y = absval(x)
y = x;
if y < 0
  y = -y;
end
```

Η εντολή **elseif** χρησιμοποιείται όταν υπάρχουν παραπάνω από δυο εναλλακτικές προτάσεις που θα πρέπει να ξεχωρίσουμε σε μια αλληλουχία δοκιμών σωστό/λάθος. Για παράδειγμα θα φτιάξουμε ένα M-script file που υπολογίζει την συνάρτηση

$$\mathrm{sgn}(x)=\begin{cases} 1 & \text{αν } x>0, \\ 0 & \text{αν } x=0, \\ -1 & \text{αν } x<0. \end{cases}$$

```
function y = signum(x)
if x > 0
    y = 1;
elseif x == 0
    y = 0;
else
   y = -1;
end
```

Λογικές εκφράσεις

help relop

για να δούμε στο Matlab όλες τις σχέσεις. Μερικοί από τους λογικούς τελεστές είναι οι **&** (AND) και **|** (OR) και ~ (NOT). Για παράδειγμα η έκφραση (x > 0) | (y > 0) θα ελέγξει αν το x ή το y (ή και τα δύο) είναι θετικοί.

- Σε ένα κλάδο if ή elseif αντί για μια πρόταση που μπορεί να ελεγχεί αριθμητικά μπορούμε να θέσουμε λογικές εκφράσεις. Π.χ.

```
>> 2 > 3

ans =

0
```

Το Matlab θέτει την τιμή 0 (λάθος) και 1 (σωστό) **help logical...**

```
>> 2 | 3

ans =

    1
```

Ο τελεστής OR | δίνει την απάντηση 0 αν και τα δυο ορίσματα είναι μηδέν και 1 αλλιώς.

Αν ο έλεγχος γίνεται σε πίνακες τότε αυτός γίνεται κατά συνιστώσες

```
>> [2 3] < [3 2]

ans =

    1     0

>> x = -2:2; x >= 0

ans =

    0     0     1     1     1
```

Μπορούμε να επιλέξουμε ορισμένα στοιχεία ενός πίνακα που ικανοποιούν μια συγκεκριμένη συνθήκη. Για παράδειγμα

```
>> x(x >= 0)

ans =

    0     1     2
```

Το Matlab έχει επιπλέον τις εντολές **any** και **all** που ελέγχουν αν κάποιο στοιχείο ικανοποιεί κάποια συνθήκη **if any(x == 0);...; end** Όμοια για την εντολή **all**

Παραδείγματα λογικών εκφράσεων και κλάδων

Θέλουμε να φτιάξουμε ένα M-script file που να ορίζει την συνάρτηση

$$f(x) = \begin{cases} \sin(x)/x & \text{αν } x \neq 0 \\ 1 & \text{αν } x = 0 \end{cases}$$

Αυτό μπορεί να επιτευχθεί με το παρακάτω:

```
function y = f(x)
if x == 0
    y = 1;
else
    y = sin(x)/x;
end
```

Το script θα δουλέψει άν εισαγάγουμε x βαθμωτό αλλά όχι άν το x είναι διάνυσμα ή πίνακας. Θα μπορούσαμε να βάλουμε **./** αντί για / στον δεύτερο ορισμό αλλά αν το x διάνυσμα είχε μηδενικά και μη μηδενικά στοιχεία το Matlab θα στο **if** θα είχε αποτέλεσμα **False** και θα προχωρούσε στον δεύτερο ορισμό. Έτσι στην απάντηση θα είχαμε μερικά στοιχεία ως **NaN** (not a number) επειδή το 0/0 είναι ακαθόριστη ποσότητα.

Μια λύση στο πρόβλημα είναι η εξής:

```
function y = f(x)
y = ones(size(x));
for n = 1:prod(size(x))
    if x(n) ~= 0
        y(n) = sin(x(n))/x(n);
    end
end
```

Εναλλακτικά θα μπορούσαμε να έχουμε

```
function y = f(x)
if x ~= 0
    y = sin(x)./x;
    return
end
y = ones(size(x));
for n = 1:prod(size(x))
    if x(n) ~= 0
        y(n) = sin(x(n))/x(n);
    end
end
```

Πολλές φορές τα loops και οι κλάδοι μπορούν να αποφευχθούν με την χρήση λογικών διανυσμάτων. Έχουν το πλεονέκτημα να είναι πιο συνοπτικοί και ικανοί για να τρέχουν script όπως το προηγούμενο.

```
function y = f(x)
y = ones(size(x));
n = (x ~= 0);
y(n) = sin(x(n))./x(n);
```

Το **n** είναι ένα λογικό διάνυσμα ίδιας διάστασης με το x με 1 σε κάθε στοιχείο που έχει μη-μηδενικά στοιχεία και 0 αλλιώς. Οπότε το y(n) ξαναορίζει μόνο όσα στοιχεία του y που έχουν μη-μηδενικά στοιχεία και αφήνει αμετάβλητα τα υπόλοιπα στοιχεία ίσα με 1. Αν έχουμε ένα πίνακα με 10.000 στοιχεία το παραπάνω script είναι πολύ γρηγορότερο από ένα script με loops.

Κλάδοι με την εντολή switch

Με την εντολή **switch** μπορούμε να κάνουμε κλάδους με διάφορες συνθήκες με ισότητες, για παράδειγμα

```
function y = count(x)
switch x
case 1
    y = 'one';
case 2
    y = 'two';
otherwise
    y = 'many';
end
```

Η switch υπολογίζει το x και μετά εκτελείται το script και πηγαίνει σε όποια **case** έχει την ίδια τιμή το x.

Το swicth **δεν** επιτρέπει εκφράσεις με διανύσματα, αλλά επιτρέπει εκφράσεις με strings.

Loops και η εντολή while

Θα υπολογίσουμε το άθροισμα

1 + 1/2^4 + 1/3^4+....

μέχρι όρους όπου το άθροισμα δεν αλλάζει σε σχέση με την αριθμητική ακρίβεια του υπολογιστή.

```
n = 1;

oldsum = -1;

newsum = 0;

while newsum > oldsum

    oldsum = newsum;

    newsum = newsum + n^(-4);

    n = n + 1;

end

newsum
```

Βγαίνοντας από ένα Loop

Μερικές φορές μπορέι να χρειαστεί το script να βγει από ένα βρόγχο if ή while όταν μια βοηθητική συνθήκη ικανοποιείται. Αυτό επιτυγχάνεται με την εντολή **break** όπου το Matlab σταματά να τρέχει το loop και πηγαίνει στην επόμενη εντολή από το end του loop. Για παράδειγμα

```
newsum = 0;
for n = 1:100000
    oldsum = newsum;
    newsum = newsum + n^(-4);
    if newsum == oldsum
        break
    end
end
newsum
```

Άλλες εντολές προγραμματισμού

Subfunction. Μπορούμε να χρησιμοποιήσουμε την εντολή function μετά από την εντολή function που εμφανίζεται στην αρχή του ορισμού μιας συνάρτησης. Η εντολή αυτή δεν είναι προσβάσιμη από το command line. Για παράδειγμα, η παρακάτω συνάρτηση υπολογίζει το άθροισμα των κυβικών ριζών ενός διανύσματος x πραγματικών αριθμών.

```
function y = sumcuberoots(x)

y = sum(cuberoot(x));

% ---- Subfunction starts here.

function z = cuberoot(x)

z = sign(x).*abs(x).^(1/3);
```

Η cuberoot δίνει την κυβικές ρίζες του διανύσματος x κατά συνιστώσες αλλά δεν είναι προσβάσιμη από το command line. Μπορούμε να χρησιμοποιούμε subfunctions **μόνο** σε function M-files και όχι σε script M-files.

- >>edit ezplot (έχει 3 διαφορετικές subfunctions)

Εντολές για input output

Στο Matlab μπορούμε να χρησιμοποιήσουμε στα M-files τις εντολές **nargin, nargout, varargin,** και **varargout** για να διαχειριστούμε τις αριθμητικές μεταβλητές εισόδου/εξόδου. Για διαφορετικού τύπου μεταβλητές χρησιμοποιούμε τις **isnumeric** και **ischar**

```
function s = add(x, y, z)
if nargin < 2
    error('At least two input arguments are required.')
end
if nargin == 2
    s = x + y;
else
    s = x + y + z;
end
```

Αν είχαμε πολλές μεταβλητές ακαθόριστου πλήθους τότε ενδείκνυται η **varargin**

```
function s = add(varargin)
s = sum([varargin{:}]);
```

Στο προηγούμενο παράδειγμα αν οι μεταβλητές εισόδου δεν είναι αριθμοί το Matlab θα τις προσθέσει και το πιθανότερο είναι να δώσει σαν αποτέλεσμα κάτι χωρίς νόημα. Μπορούμε να ελέγξουμε αν οι μεταβλητές εισόδου είναι αριθμοί με το παρακάτω έλεγχο if

```
function s = add(varargin)

if ~isnumeric([varargin{:}])

error('Inputs must be floating point numbers.')

end

s = sum([varargin{:}]);
```

- Αν οι μεταβλητές εξόδου είναι λιγότερες από αυτές που έχουν δηλωθεί στο function M-file τότε η συνάρτηση καλείται και απλά δεν καταχωρούνται οι υπόλοιπες μεταβλητές εξόδου. Π.χ

```
function [x, y] = rectangular(r, theta)

x = r.*cos(theta);

y = r.*sin(theta);
```

Αν τρέξουμε rectangular(2,1) θα πάρουμε σαν αποτέλεσμα μόνο την συντεταγμένη x. Θα πρέπει να δώσουμε [x,y]=rectangular(2,1) για να πάρουμε και τις δυο x,y. Στο παρακάτω script κάνουμε μια τροποποίηση ώστε να πάρουμε το αποτέλεσμα σαν ένα μιγαδικό αριθμό

```
function [x, y] = rectangular(r, theta)

x = r.*cos(theta);

y = r.*sin(theta);

if nargout < 2

x = x + i*y;

end
```

Input output από τον χρήστη στην οθόνη

Warnings και errors. Με την εντολή **error** το script τερματίζεται ενώ με την εντολή **warning** ή **disp** το script συνεχίζει να τρέχει και απλά τυπώνει ένα μήνυμα όταν το M-file έρθει αντιμέτωπο με ένα πρόβλημα το οποίο μπορεί να επηρεάσει την εγκυρότητα του αποτελέσματος αλλά δεν είναι αναγκαστικά ουσιώδες.

Μπορούμε να χρησιμοποιήσουμε την εντολή **disp** για να εμφανισθούν στην οθόνη μηνύματα κειμένου ή με αριθμητικό αποτέλεσμα. Π.χ.

```
>> x = 2 + 2; disp(['The answer is ' num2str(x) '.'])
```

- Είχαμε δει επίσης της εντολές **pause, keyboard, input.** Με το pause διακόπτεται προσωρινά το M-file και τρέχει όταν πατήσουμε ένα οποιοδήποτε πλήκτρο. Με την εντολή keyboard διακόπτεται το M-file και μας δίνεται η δυνατότητα να έχουμε αλληλεπίδραση με ένα άλλο prompt όπου μπορούμε να αλλάξουμε τις μεταβλητές. Με την εντολή **return** το M-file επανέρχεται σε λειτουργία. Τέλος με τη εντολή **input** δίνεται η δυνατότητα στον χρήστη να εισάγει δεδομένα για το πρόγραμμα σε μια γραμμή.

Για παράδειγμα:

```
answer = input([’Algorithm is converging slowly; ’, ...
’continue (yes/no)? ’], ’s’);
if ~isequal(answer, ’yes’)
return
end
```

Το δεύτερο όρισμα **'s'** στο **input** οδηγεί το Matlab στο να μην υπολογίσει την απάντηση που θα εισάγει ο χρήστης αλλά να το καταχωρήσει σε μια μεταβλητή answer. Μετά η isequal θα συγκρίνει το αποτέλεσμα με το string 'yes'. Στο παράδειγμα αποφασίσαμε να ότι αν ο χρήστης οτιδήποτε άλλο εκτός από την πλήρη λέξη yes το M-file θα τερματίσει. Άλλη προσέγγιση θα ήταν να συγκρίνουμε το πρώτο γράμμα στο answer δηλ answer(1) με το γράμμα 'y' για να σταματήσει το M-file αν η απάντηση του χρήστη είναι 'no'.

Στην παρακάτω συνάρτηση αν μια γραφική παράσταση είναι ενεργή χρησιμοποιούμε την εντολή **ginput** για να πάρουμε τις συντεταγμένες ενός σημείου που ο χρήστης επιλέγει με το ποντίκι.

```
function xmarksthespot
if isempty(get(0, 'CurrentFigure'))
    error('No current figure.')
end
flag = ~ishold;
if flag
    hold on
end
disp('Click on the point where you want to plot an X.')
[x, y] = ginput(1);
plot(x, y, 'xk')
if flag
    hold off
end
```

Υπολογισμός

eval και **feval.** Με τη **eval** μπορούμε να υπολογίσουμε μια μεταβλητή η οποία είναι στην μορφή string. Π.χ.

>>eval('cos(1)')

Γενικά η **eval** χρησιμοποιείται σε ένα M-file όπου οι μεταβλητές σχηματίζονται από strings που περιέχουν εντολές.

- Με την **feval** μπορούμε να υπολογίσουμε την τιμή μιας συνάρτησης την οποία δηλώνουμε με ένα string. Π.χ

  >>feval(’atan’, 1) ή >>feval(@atan, 1) παίρνουμε την τιμή της συνάρτησης atan στο 1.

```
function final = iterate(func, init, num)

final = init;

for k = 1:num

final = feval(func, final);

end
```