ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ
ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΠΟΛΙΤΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΔΟΜΟΣΤΑΤΙΚΗΣ

Ακαδ. Έτος: 2018-2019

Μάθημα: Μέθοδοι Επίλυσης με Η/Υ

Τετάρτη, 21/11/2018

Διδάσκοντες: Ν.Δ. Λαγαρός (Αν. Καθηγητής), Α. Στάμος (ΕΔΙΠ), Χ. Φραγκουδάκης (ΕΔΙΠ)
Αμβ. Σαββίδης

Παραδείγματα για την 6η παράδοση - Συναρτήσεις στο MATLAB

1. Υπολογισμός πλήθους οπτόπλινθων

Το πλήθος οπτόπλινθων (τούβλων) μήκους Lt που χρειάζονται για να καλύψουν καθαρό μήκος (χωρίς υποστυλώματα, δοκούς, πλάκες) La σε μία στρώση με πάχος συνδετικού κονιάματος Lk μεταξύ των οπτόπλινθων και μεταξύ οπτόπλινθων και σκυροδέματος, είναι:

$$k = f(La, Lt, Lk) = \left\{ \frac{La - Lk}{Lt + Lk} \right\}$$

όπου τα άγκιστρα σημαίνουν στρογγυλοποίηση προς τα πάνω.

Για να καλυφθεί δρομικός τοίχος μήκους καθαρού μήκους L σε μία στρώση το πλήθος των οπτόπλινθων k δίνεται από τον τύπο με La=L, Lk=0.01m και Lt=0.19m. Το πλήθος των στρώσεων Ns δίνεται από τον ίδιο τύπο με La=h, Lk=0.01m και Lt=0.06m. Το πλήθος N όλων των οπτόπλινθων είναι:

Να συνταχθεί συνάρτηση σε Matlab η οποία με δεδομένα τις καθαρές διαστάσεις κάτοψης Lx, Ly ενός ορόφου και του καθαρού του ύψους h, να επιστρέφει το πλήθος Nor των οπτόπλινθων που απαιτούνται.

Να γίνει το ίδιο για το τελευταίο όροφο όπου μόνο οι 2 διαδοχικές του πλευρές καλύπτονται με οπτόπλινθους (οι υπόλοιπες καλύπτονται με γυαλί).

Λύση με υποσυναρτήσεις

Καταρχήν θα συνταχθεί υποσυνάρτηση που θα υλοποιεί τον παραπάνω τύπο. Από την βοήθεια του Matlab (ή από μία μηχανή αναζήτησης όπως τη Google) βρίσκουμε ότι η έτοιμη συνάρτηση ceil() του Matlab κάνει στρογγυλοποίηση στον αμέσως μεγαλύτερο ακέραιο. Ο κώδικας είναι:

```
function k=toybla(La, Lt, Lk)
    % Βρίσκει το πλήθος τούβλων μήκους Lt που χωρούν σε μήκος τοίχου La
    % με συνδετικό κονίαμα πάχους Lk.
    k = (La-Lk)./(Lt+Lk);
    k = ceil(k);
end
```

Στη συνέχεια συντάσσεται υποσυνάρτηση που υπολογίζει το πλήθος οπτόπλινθων n για τοίχο καθαρού μήκους La και καθαρού ύψους h, με τη βοήθεια της προηγούμενης υποσυνάρτησης. Ο κώδικας είναι:

```
function n=toixos(La, h)
    % Finds the number of bricks of a rectangular wall of length L
    % and height h.
    n1 = toybla(La, 0.19, 0.01);    %Πλήθος τούβλων σε 1 στρώση
    nstro = toybla(h, 0.06, 0.01);  %Πλήθος στρώσεων που καλύπτουν τοίχο
                                    %ύψους h
```

```
    n = nstro.*n1;
end
```

Ας σημειωθεί ότι η μεταβλητή La της (υπο)συνάρτησης toybla() είναι τελείως διαφορετική από τη μεταβλητή La της (υπο)συνάρτησης toixos(). Μπορεί να έχουν το ίδιο όνομα αλλά έχουν διαφορετικό «επώνυμο»: η πρώτη έχει «επώνυμο» toybla και δεύτερη έχει «επώνυμο» toixos. Στην πρώτη κλήση της συνάρτησης στις μεταβλητές La, Lt, Lk της συνάρτησης toybla() αντιγράφονται αντίστοιχα η τιμή της μεταβλητής La της συνάρτησης toixos(), και οι τιμές 0.19 και 0.01. Στην δεύτερη κλήση της συνάρτησης στις μεταβλητές La, Lt, Lk της συνάρτησης toybla() αντιγράφονται αντίστοιχα η τιμή της μεταβλητής h της συνάρτησης toixos(), και οι τιμές 0.06 και 0.01.

Τέλος συντάσσεται συνάρτηση (όχι υποσυνάρτηση) που υπολογίζει το πλήθος οπτόπλινθων για τους 4 τοίχους ενός ορόφου καθαρών διαστάσεων Lx, Ly και καθαρού ύψους h, με τη βοήθεια της προηγούμενης υποσυνάρτησης. Ο κώδικας μαζί με τις υποσυναρτήσεις είναι:

```
function nor = orofos(Lx, Ly, h)
    %Βρίσκει το πλήθος των τούβλων σε ορθογωνικό όροφο Lx x Ly  και
    %ύψους h
    nor = 2.*(toixos(Lx, h) + toixos(Ly, h));
end
function n=toixos(La, h)
    % Finds the number of bricks of a rectangular wall of length L
    % and height h.
    n1 = toybla(La, 0.19, 0.01);    %Πλήθος τούβλων σε 1 στρώση
    nstro = toybla(h, 0.06, 0.01);  %Πλήθος στρώσεων που καλύπτουν τοίχο
                                    %ύψους h
    n = nstro.*n1;
end
function k=toybla(La, Lt, Lk)
    % Βρίσκει το πλήθος τούβλων μήκους Lt που χωρούν σε μήκος τοίχου La
    % με συνδετικό κονίαμα πάχους Lk.
    k = (La-Lk)./(Lt+Lk);
    k = ceil(k);
end
```

Ας σημειωθεί ότι η συνάρτηση orofos() και όλες οι υποσυναρτήσεις της γράφονται στο ίδιο αρχείο orofos.m, και ότι η συνάρτηση orofos() πρέπει να γραφεί πριν από όλες τις υποσυναρτήσεις.

<u>Ελέγχουμε το πρόγραμμα με το script:</u>

```
orofos(10, 8, 2.8)
      ans = 7200
```

<u>Λύση με συναρτήσεις και υποσυναρτήσεις</u>

Για το δεύτερο ζητούμενο για τον τελευταίο όροφο, συντάσσεται νέα συνάρτηση παρόμοια με την προηγούμενη. Ο κώδικας είναι:

```
function nort = ortel(Lx, Ly, h)
    %Βρίσκει το πλήθος των τούβλων σε ορθογωνικό όροφο Lx x Ly
    % και ύψους h για 2 τοίχους.
    nort = toixos(Lx, h)+toixos(Ly, h);
end
```

Όμως έλεγχος της συνάρτησης με το παρακάτω script δίνει μήνυμα λάθους:

```
ortel(10, 8, 2.8)
      error: 'toixos' undefined near line 4 column 12
```

Αυτό οφείλεται ότι η υποσυνάρτηση toixos() της συνάρτησης orofos() μπορεί να προσπελαστεί από τη συνάρτηση orofos() και τις υποσυναρτήσεις της, αλλά από κανέναν άλλον. Για να διορθωθεί το πρόβλημα δοκιμάζουμε να μετατρέψουμε την συνάρτηση ortel() σε υποσυνάρτηση της orofos()

(δηλαδή να γράψουμε την ortel() μέσα στο αρχείο orofos.m). Όμως πάλι το script δίνει μήνυμα λάθους:

```
ortel(10, 8, 2.8)
      error: 'ortel' undefined near line 1 column 1
```

Αυτό οφείλεται ότι επειδή τώρα η ortel() είναι της συνάρτησης orofos(), μπορεί να προσπελαστεί μόνο από τη συνάρτηση orofos() και τις υποσυναρτήσεις της, αλλά όχι από το script.
Η ορθή λύση είναι να μετατρέψουμε την υποσυνάρτηση toixos() σε κανονική συνάρτηση (στο δικό της αρχείο toixos.m, μαζί με την υποσυνάρτηση toybla()). Τώρα το script λειτουργεί κανονικά:

```
orofos(10, 8, 2.8)
      ans = 3600
```

Με τα παραπάνω τίθεται το ερώτημα γιατί να υπάρχουν υποσυναρτήσεις, που θα απαντηθεί στο επόμενο παράδειγμα.

2. Συναρτήσεις ημίτονο και συνημίτονο με σειρές

Να συνταχθούν οι συναρτήσεις mysin() και mycos() που να υπολογίζουν το ημίτονο και συνημίτονο (σε μοίρες) χρησιμοποιώντας του 4 πρώτους όρους από τη σειρά Taylor:

$$\sin(x) = x + \sum_{k=1}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$$

$$\cos(x) = 1 + \sum_{k=1}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}$$

Ας σημειωθεί ότι οι γωνίες x στις σειρές Taylor είναι σε ακτίνια (rad).

Λύση ημιτόνου με υποσυναρτήσεις

Εφόσον οι όροι της σειράς είναι εφαρμογή του ίδιου τύπου για διαφορετικό k, ο τύπος αυτός θα υλοποιηθεί σε υποσυνάρτηση, που θα έχει όρισμα εισόδου το k και θα επιστρέφει τον όρο. Από την βοήθεια του Matlab (ή από μία μηχανή αναζήτησης όπως τη Google) βρίσκουμε ότι η έτοιμη συνάρτηση factorial() του Matlab υπολογίζει το παραγοντικό. Ο κώδικας είναι:

```
function z=oros(k)    % ΕΧΕΙ ΣΦΑΛΜΑ
    % Υπολογίζει έναν όρο της σειράς.
    kk = 2.*k+1;
    z = (-1).^k .* x.^kk/factorial(kk);
end
```

Για να υπολογιστεί το ημίτονο μετατρέπουμε τη γωνία σε rad και καλούμε την προηγούμενη υποσυνάρτηση 4 φορές. Από την βοήθεια του Matlab (ή από μία μηχανή αναζήτησης όπως τη Google) βρίσκουμε ότι η έτοιμη σταθερά pi του Matlab έχει ως τιμή τον αριθμό π. Ο κώδικας είναι:

```
function y=mysin(x)   % ΕΧΕΙ ΣΦΑΛΜΑ
    % Υπολογίζει το ημίτονο με 4 όρους της σειράς.
    x = x.*pi./180;
    y = x + oros(1) + oros(2) + oros(3) + oros(4);
end
```

Όμως έλεγχος της συνάρτησης με το παρακάτω script δίνει μήνυμα λάθους:

```
mysin(60)
      error: 'x' undefined near line 10 column 20
```

Αυτό οφείλεται ότι η υποσυνάρτηση oros() δεν έχει πρόσβαση στη μεταβλητή x της συνάρτησης mysin(x) (ούτε και σε οποιασδήποτε άλλης μεταβλητής οποιασδήποτε άλλης συνάρτησης ή

υποσυνάρτησης). Συνεπώς η μεταβλητή x πρέπει να είναι όρισμα εισόδου της υποσυνάρτησης oros(). Διόρθωση του κώδικα δίνει:

```
function y=mysin(x)
    % Υπολογίζει το ημίτονο με 4 όρους της σειράς.
    x = x.*pi./180;
    y = x + oros(x, 1) + oros(x, 2) + oros(x, 3) + oros(x, 4);
end

function z=oros(x, k)
    % Υπολογίζει έναν όρο της σειράς.
    kk = 2.*k+1;
    z = (-1).^k .* x.^kk/factorial(kk);
end
```

Έλεγχος της συνάρτησης με το παρακάτω script δίνει σωστό αποτέλεσμα:

```
mysin(30)
      ans =   0.50000
```

Λύση συνημιτόνου με υποσυναρτήσεις

Ακολουθώντας τον ίδιο συλλογισμό ο υπολογισμός του συνημιτόνου δίνεται από τον κώδικα:

```
function y=mycos(x)
    % Υπολογίζει το συνημίτονο με 4 όρους της σειράς.
    x = x.*pi./180;
    y = 1 + oros(x, 1) + oros(x, 2) + oros(x, 3) + oros(x, 4);
end

function z=oros(x, k)
    % Υπολογίζει έναν όρο της σειράς.
    kk = 2*k;
    z = (-1).^k * x.^kk/factorial(kk);
end
```

Έλεγχος της συνάρτησης με το παρακάτω script δίνει σωστό αποτέλεσμα:

```
mycos(30)
      ans =   0.86603
```

Αιτιολόγηση υποσυναρτήσεων

Παρατηρούμε ότι και η συνάρτηση mysin() και η mycos() έχουν υποσυνάρτηση με όνομα oros(). Οι δύο υποσυναρτήσεις oros() είναι διαφορετικές μεταξύ τους. Γιατί δεν συγχέονται; Διότι η συνάρτηση mysin() δεν μπορεί να προσπελάσει τις υποσυναρτήσεις άλλης συνάρτησης (της mycos()), και έτσι αναγκαστικά όταν καλεί την oros() καλεί τη δικιά της υποσυνάρτηση oros(). Ομοίως για την mycos().

Είναι σχεδόν σίγουρο ότι σε μεγάλα προγράμματα θα υπάρχουν συναρτήσεις διαφορετικές με το ίδιο όνομα, ιδίως αν το πρόγραμμα αναπτύσσεται από πολλούς διαφορετικούς προγραμματιστές. Η χρήση υποσυναρτήσεων εξαλείφει σε μεγάλο βαθμό αυτό το πρόβλημα. Συνεπώς όπου είναι δυνατόν (δηλαδή όταν μία συνάρτηση δεν καλείται από άλλο μέρος του προγράμματος) θα πρέπει να γίνεται χρήση υποσυναρτήσεων αντί για συναρτήσεις.

Αιτιολόγηση τοπικών μεταβλητών

Παρατηρούμε ότι το παρακάτω script λειτουργεί όπως πρέπει και δίνει σωστά αποτελέσματα:

```
clear; clc; close all;
x = 30;
mysin(x)
mycos(x)
```

x

Τρέξιμο του script δίνει:

```
ans =  0.50000
ans =  0.86603
x =  30
```

Κοιτάζοντας τον κώδικα της mysin() και της mycos() βλέπουμε ότι αλλάζουν την τιμή της μεταβλητής x (τη μετατρέπουν σε ακτίνια). Παρόλα αυτά το script τυπώνει την τιμή 30 για τη μεταβλητή x. Αυτό συμβαίνει διότι η μεταβλητή x της συνάρτησης mysin() είναι τελείως διαφορετική από τη μεταβλητή x του script και από τη μεταβλητή x της συνάρτησης mycos() (έχει διαφορετικό «επώνυμο»). Συνεπώς όταν αλλάζει η τιμή της δεν αλλάζουν οι τιμές των άλλων μεταβλητών. Αυτό έχει γίνει έτσι ώστε η κλήση μίας συνάρτησης να μην έχει παρενέργειες (side effects) στην λειτουργία μίας άλλης και πιθανότατα άσχετης συνάρτησης.

Λύση ημιτόνου με εμφωλευμένες υποσυναρτήσεις

Μερικές φορές, όχι συχνά, θα ήταν χρήσιμο μία συνάρτηση να έχει πρόσβαση στις μεταβλητές μίας άλλης. Για παράδειγμα, η πρώτη έκδοση της υποσυνάρτησης mysin() (που έχει σφάλμα) είναι πιο ευανάγνωστη από τη δεύτερη έκδοση. Αν όμως η υποσυνάρτηση oros() είχε πρόσβαση στις μεταβλητές της συνάρτησης mysin(), η πρώτη (ευανάγνωστη) έκδοση δεν θα είχε σφάλμα.

Η πρόσβαση μπορεί να υπάρχει αν η υποσυνάρτηση oros() γραφεί μέσα στη συνάρτηση mysin(), δηλαδή αν είναι εμφωλευμένη στη συνάρτηση mysin(). Ο σχετικός κώδικας γίνεται:

```
function y=mysine(x)
    % Υπολογίζει το ημίτονο με 4 όρους της σειράς.

    function z=oros(k)
        % Υπολογίζει έναν όρο της σειράς.
        kk = 2*k+1;
        z = (-1).^k * x.^kk/factorial(kk);
    end

y = x + oros(1) + oros(2) + oros(3) + oros(4);
end
```

Η εμφωλευμένη υποσυνάρτηση είναι μία υποσυνάρτηση που έχει πρόσβαση στις μεταβλητές της συνάρτησης στην οποία είναι εμφωλευμένη. Δεν έχει πρόσβαση στις μεταβλητές άλλων συναρτήσεων ή υποσυναρτήσεων, όπως ακριβώς και οι κανονικές υποσυναρτήσεις.