Λογικά Διανύσματα

Τα λογικά διανύσματα του Matlab είναι πολύ χρήσιμα εργαλεία. Για παράδειγμα ας υποθέσουμε ότι θέλουμε να κάνουμε την γραφική παράσταση της tan(x) στο διάστημα από -3π/2 μέχρι 3π/2.

```
>>x = -3/2*pi : pi/100 : 3/2*pi;
>>y = tan(x);
>>plot(x, y)
```

Το πρόβλημα είναι ότι οι τιμές της συνάρτησης απειρίζονται (θετικά ή αρνητικά) όταν το x πλησιάζει ακέραια πολλαπλάσια του π/2. Αν όμως διώξουμε τις μεγάλες τιμές της συνάρτησης μπορούμε να έχουμε μια εικόνα της γραφικής της παράστασης.

```
>>x = -3/2*pi : pi/100 : 3/2*pi;
>>y = tan(x);
>>y = y .* (abs(y) < 1e10);
>>plot(x, y)
```

Μετρώντας τυχαίους αριθμούς

Η συνάρτηση **rand** δίνει έναν τυχαίο πραγματικό αριθμό στο διάστημα [0,1] από την ομοιόμορφη κατανομή. Η εντολή **rand(1 , n)** δίνει ένα διάνυσμα γραμμή με n τέτοιους αριθμούς.

- Ας βρούμε ένα διάνυσμα με τέτοιους αριθμούς.

  >>r = rand(1,7)

  κι ας ελέγξουμε ότι ένα η εντολή **r < 0.5** δίνει το σωστό λογικό διάνυσμα.

- Χρησιμοποιώντας την συνάρτηση **sum** στην λογική έκφραση r < 0.5 θα μας μετρήσει το πλήθος των στοιχείων του r που είναι < 0.5

  >>sum(r < 0.5)

- Τώρα με ανάλογο τρόπο μετρήστε τα στοιχεία του r που είναι >=0.5

  Επειδή οι τυχαίοι αριθμοί παράγονται από την ομοιόμορφη κατανομή θα περιμέναμε το πλήθος των αριθμών που είναι <0.5 να γίνονται οι μισοί καθώς παίρνουμε ολοένα περισσότερους τυχαίους αριθμούς. Δοκιμάστε το.

  Δεν θα πάρουμε ποτέ το ίδιο αποτέλεσμα!!

Ας λύσουμε το προηγούμενο πρόβλημα δίχως την χρήση των λογικών διανυσμάτων, κι ας μετρήσουμε τον χρόνο που κάνει το Matlab να το υλοποιήσει

```
tic                         % start
a = 0;                      % a will be a number >= 0.5
b = 0;                      % b < 0.5
for n = 1:5000
    r = rand;
    if r >= 0.5             % generate one number per loop
      a = a + 1;
    else
     b = b + 1;
    end;
end;
t = toc;                    % finish
disp( ['less than 0.5: ' num2str(a)] )
disp( ['time: ' num2str(t)] )
```

Ρίψη ζαριών

Όταν ρίχνουμε ένα ζάρι είναι εξίσου πιθανό να έρθει σαν αποτέλεσμα ένας από τους αριθμούς 1,2...6. Οπότε, αν α rand είναι ένας τυχαίος αριθμός στο [0 , 1) , 6 * rand θα είναι ένας αριθμός στο [0 , 6) και 6 * rand + 1 θα είναι ένας τυχαίος αριθμός στο [1 , 7). Αν με την εντολή floor δεν δίνουμε σημασία στα δεκαδικά ψηφία τότε έχουμε έναν τυχαίο αριθμό 1,2,.. 6.

- Να παραχθεί ένα διάνυσμα d με 20 τυχαίους ακεραιους από 1 μέχρι 6

  >>d = floor(6 * rand(1, 20)) + 1

- Μετρήστε το πλήθος των “εξαριών” με λογική διάνυσμα d==6.
- Επαληθεύστε το αποτέλεσμα
- Εκτιμήστε την πιθανότητα που ήρθε το 6 διαιρώντας με το 20.
- Επαναλάβετε το πείραμα με μερισσότερους αριθμούς. Όσο περισσότερους αριθμούς παίρνουμε τόσο η πιθανότητα να έρθει 6 πλησιάζει την θεωρητική τιμή 0.1667, 1/6.

Δείκτες με λογικά διανύσματα

Τα στοιχεία ενός διανύσματος αναφέρονται με δείκτες και οι δείκτες είναι διανύσματα

```
>>a = [-2 0 1 5 9];

>>a([5 1 3])

9    -2   1
```

- Γενικά αν x και v είναι διανύσματα και το v έχει n στοιχεία, τότε x(v) σημαίνει

  [x(v(1)), x(v(2)), … , x(v(n))].

```
>>a(logical([1 1 1 0 0]))                    % Τι θα δώσει??

>>(logical([0 0 0 0 0]))
```

- Ένα λογικό διάνυσμα μπορεί να είναι δείκτες ενός άλλου διανύσματος
- Μόνο τα στοιχεία του x που αντιστοιχούν στα 1, επιστρέφονται
- Τα x και v πρέπει να έχουν τις ίδιες διαστάσεις.

Τα λογικά διανύσματα είναι ένας εύκολος τρόπος να σβήνουμε ορισμένα στοιχεία από ένα διάνυσμα

```
>>a = a(a > 0)
```

Η εντολή αυτή σβήνει όλα τα μη-θετικά στοιχεία του διανύσματος a γιατί η εντολή a > 0 έχει ως αποτέλεσμα το λογικό διάνυσμα [0 0 1 1 1]. Πως μπορούμε να είμαστε σίγουροι ότι πράγματι πρόκειται για λογική έκφραση?

```
>>islogical(a > 0)
```

απαντά 1.

- Το διάνυσμα [0 0 1 1 1] δεν είναι λογικό γιατί η ερώτηση

```
>>islogical([0 0 1 1 1])
```

δίνει 0.

Λογικές συναρτήσεις

Το Matlab έχει ένα πλήθος χρήσιμες λογικές συναρτήσεις που δρουν σε βαθμωτά, διανύσματα ή πίκακες.

- any(x) δίνει 1 αν κάποιο από τα στοιχεία του x είναι μη-μηδενικό
- all(x) δίνει 1 αν όλα τα στοιχεία του x είναι μη-μηδενικά
- Exist('a') δίνει 1 αν a είναι μια ενεργή μεταβλητή στο workspace.
- find(x) δίνει ένα διάνυσμα που περιέχει τους δείκτες με τα μή-μηδενικά στοιχεία του x.

```
>>x = [8 1 -4 8 6];

>>find(x)

>>find(x >= max(x))
```

- isempty(x) δίνει 1 αν το x είναι ένα κενό διάνυσμα ή πίνακας και 0 αλλιώς
- isinf(x) δίνει 1 για τα στοιχεία του x που είναι +Inf ή -Inf
- isnan(x) δίνει 1 για τα στοιχεία του x που είναι NaN.

Χρήση των any και all

Επειδή οι εντολές any και all με ορίσματα διανύσματα επιστρέφουν βαθμωτά είναι ιδιαίτερα χρήσιμες σε ελέγχους με if.

```
if all( a >= 1 )
  κάνε κάτι
end
```

σημαίνει αν όλα τα στοιχεία του διανύσματος a είναι >=1, τότε κάνε κάτι. Αν θέλαμε να κάνουμε κάτι αν δυο διανύσματα είναι ίσα

```
if a == b
   πρόταση
end
```

ομως είναι λάθος να πούμε το εξής

```
if a~= b          %ΛΑΘΟΣ ΛΑΘΟΣ ΛΑΘΟΣ
   πρόταση
end
```

Το σωστό είναι

```
if any(a~= b)
   πρόταση
end
```

Λογικά διανύσματα αντί βρόγχων elseif

Ας θεωρήσουμε το παρακάτω παράδειγμα.

Φορολογητέο εισόδημα φόρος

10 000 ή λιγότερο 10% του φορ. Εισοδήματος

ανάμεσα 10 000 και 1000 + 20% του ποσού που το φορ. 20 000 Εισόδημα ξεπερνά τις 10 000

πάνω από 20 000 3 000 + 50% του ποσού που το φορ. Εισ. ξεπερνά τις 20 000.

Π.χ. Φορός που αντιστοιχεί σε φορολογητέο εισόδημα 30 000 είναι

3 000 + 50 % (30 000 – 20 000) δηλ 8 000.

θέλουμε να υπολογίσουμε τον φόρο των παρακάτω φορολογητέων εισοδημάτων 5 000, 10 000, 15 000, 30 000, 50 00.

- Ο συμβατικός τρόπος είναι να φτιάξουμε ένα πρόγραμμα που να έχει τα φορ. Εισ. Των οποίων θέλουμε να υπολογίσουμε το φόρο και να χρησιμοποιήσουμε ένα βρόγχο με elseif όπου θα ελέγχουμε κάθε περίπτωση

```
% φόρος με κλασικό τρόπο
inc = [5000 10000 15000 30000 50000];
for ti = inc
    if ti < 10000
       tax = 0.1 * ti;
    elseif ti < 20000
       tax = 1000 + 0.2 * (ti - 10000);
    else
      tax = 3000 + 0.5 * (ti - 20000);
    end;
disp( [ti tax] )
end;
```

```
     5000       500
    10000      1000
    15000      2000
    30000      8000
    50000     18000
```

- Ο “λογικός” τρόπος

```
inc = [5000 10000 15000 30000 50000];
tax = 0.1 * inc .* (inc <= 10000);
tax = tax + (inc > 10000 & inc <= 20000) ...
.* (0.2 * (inc-10000) + 1000);
tax = tax + (inc > 20000) .* (0.5 * (inc-20000) + 3000);
disp( [inc’ tax’] );
```

- Πως δουλεύει?

  Εισάγετε το inc σαν ένα διάνυσμα και δώστε

  inc <= 10000

  που δίνει το αποτέλεσμα [1 1 0 0 0]. Μετά δωστε

```
>>inc .* (inc <= 10000)
[5000 10000 0 0 0]
tax = 0.1 * inc .* (inc <= 10000)
```

Για την δεύτερη κατηγορία εισοδήματος

>>inc > 10000 & inc <= 20000

που έχει ως αποτέλεσμα [0 0 1 0 0]. Πολλαπλασιάζοντας το αποτέλεσμα με

0.2 * (inc-10000) + 1000

μας δίνει το σωστό φόρο για το εισόδημα και το αποτέλεσμα το προσθέτουμε στο tax αφού δεν πρόκειται να επηρεάσει το αποτέλεσμα (γιατι?)

Ομοίως για τα εισοδήματα >=20 000

- Ο λογικός τρόπος έχει πολύ πιο ενδιαφέρον από τον κλασσικό τρόπο.

Ασκήσεις εξοικείωσης

- Μια εταιρεία πληρώνει τους εργαζόμενούς της σε 7 διαφορετικά επίπεδα ετήσιων μισθών 12 000, 15 000, 18 000, 24 000, 35 000, 50 000, 70 000. Ο αριθμός των εργαζομένων που έχουν τους αντίστοιχους μισθούς είναι: 3000, 2500, 1500, 1000, 400, 100 και 25.

  1. Γράψτε μερικές εντολές στο command line που να υπολογίζει το μέσο όρο των μισθών (mean). (32 000)

  2. Βρείτε τον αριθμό των εργαζομένων που έχουν μισθό πάνω και κάτω από τον μέσο μισθό. Με την χρήση λογικών διανυσμάτων βρείτε ποια επίπεδα μισθών είναι πάνω και κάτω του μέσου. Πολλαπλασιάστε αυτά τα λογικά διανύσματα κατά συνιστώσες με το διάνυσμα των εργαζομένων και sum το αποτέλεσμα. (Απ. 525 πάνω 8000 κάτω)

  3. Το μέσο όρο του μισθού που κερδίζει ατομικά ένας εργαζόμενος στην εταιρεία (δηλαδή το ολικό ετήσιο μισθό διαιρεμένο με το ολικό αριθμό των εργαζομένων)

- Γράψτε εντολές στο command line με χρήση λογικών διανύσματων που να μετρά πόσα στοιχεία ενός διανύσματος x είναι αρνητικά, μηδέν ή θετικά. Ελέγξτε τα για το διάνυσμα x=[-4 0 5 -3 0 3 7 -1 6].

- Γράψτε μερικές εντολές στο command line που σβήνουν τα στοιχεία με τις μέγιστες τιμές από ένα διάνυσμα. Δοκιμάστε το για το x = [1 2 5 0 5]. Η ιδέα είναι να καταλήξουμε με [1 2 0] στο x. Να χρησιμοποιήσετε τις εντολές find και empty vector [ ].

- Ο λογαριασμός ηλεκρικού σε μια μικρή κοινότητα υπολογίζεται ως εξής

  - Αν καναναλώσει ένας κάτοικος 500 μονάδες ή λιγότερο η τιμή είναι 20 cents η μονάδα.

  - Αν η κατανάλωση είναι από 500 μέχρι 1000 μονάδες το κόστος είναι 10 ευρώ για τις πρώτες 500 μονάδες και μετά 5 cents για κάθε μονάδα πάνω από τις 500.

  - Αν η κατανάλωση είναι πάνω από 1000 μονάδες το κόστος είναι 35 ευρώ για τις πρώτες 1000 μοναδες και 10 cents για κάθε μονάδα πάνω από τις 1000.

  - Υπάρχει και ένα πάγιο 5 ευρώ ανεξάρτητα αν γίνει κατανάλωση.

  Πέντε νοικοκυριά κατανάλωσαν 200, 500, 700, 1000, 1500 σε ένα μήνα. Να γραφεί ένα script που να υπολογίζει τους λογαριασμούς τους. Να δωθούν τα αποτελέσματα σε δυο στήλες μια για τις μονάδες ηλεκτρισμού που κατανάλωσε το κάθε σπίτι και μια για το ποσό που οφείλει. (Απ. 9, 15, 25, 40, 90)

- **Προσομοίωση στο τάβλι.** Σε μια παρτίδα τάβλι οι παίχτες έριξαν συνολικά τα ζάρια 100 φορές. Να προσομοιώσετε την παρτίδα και να βρείτε πόσες φορές ήρθαν διπλές (κάθε περίπτωση ξεχωριστά), πόσες φορές άσσο-δυο και 6-5. Να θεωρήσετε ότι τα αποτελέσματα στη ρίψη των δυο ζαριών γίνεται με την εντολή

  ```
  >>d=[ floor( 6 * rand(1,100) ) + 1; floor(6 * rand(1,100) )+1];
  ```

  Να δώσετε τα αποτελέσματα σε στήλες για τις διπλές και σε ξεχωριστές γραμμές για το 1-2 και 6-5. Τα αποτελέσματα που πήρατε είναι σε συμφωνία με τα θεωρητικά αποτελέσματα της ρίψης 2 ζαριών?