4
Προγραμματισμός με το MATLAB

Δημήτριος Χριστόπουλος[1,2]

[1]Εθνικό Καποδιστριακό Πανεπιστήμιο Αθηνών, Τμήμα Οικονομικών Επιστημών
[2]dchristop@econ.uoa.gr

Άνοιξη 2011

Σημειώσεις Εργαστηρίου Γραμμικών Μαθηματικών[3].

[3]Οι ηλεκτρονικές σημειώσεις που ακολουθούν περιέχουν υπερσυνδέσεις, με ένα απλό κλικ, εσωτερικά ή εξωτερικά του κειμένου.

Περιεχόμενα

1 Προγραμματισμός με το MATLAB: συναρτήσεις και m-files

1.1 Συναρτήσεις

Υπάρχουν δύο είδη συναρτήσεων στο MATLAB. Οι συναρτήσεις με συγκεκριμένο όνομα και οι ανώνυμες συναρτήσεις, οι οποίες έχουν αναπτυχθεί στις τελευταίες εκδόσεις του MATLAB.

1.1.1 Επώνυμες συναρτήσεις

Η γενική μορφή μιας επώνυμης συνάρτησης, δηλ. μίας συνάρτησης που μπορούμε να την καλέσουμε από οποιοδήποτε κώδικα MATLAB, είναι η ακόλουθη:

```
function [out1, out2, ...] = name-of-the-function(in1, in2, ...)
```

Γράφουμε τις εντολές μας είτε με τον επεξεργαστή *editor* του MATLAB, είτε με οποιονδήποτε άλλο επεξεργαστή κειμένου ο οποίος μπορεί να σώζει το αρχείο σε οποιονδήποτε τύπο, π.χ. με το σημειωματάριο των Windows, και το σώζουμε με κατάληξη name-of-the-function.m.
Επιβεβαιώνουμε από την γραμμή πλοήγησης του MATLAB ότι εργαζόμαστε στον φάκελλο στον οποίο έχουμε αποθηκεύσει το ανωτέρω αρχείο.
Όταν κατόπιν είτε στο παράθυρο εντολών Command Window είτε μέσα σε κάποιο άλλο αρχείο γράψουμε:

```
[out1, out2, ...]=name-of-the-function(in1, in2, ...)
```

τότε θα προκύψει η απάντηση:

```
out1=...
out2=...
...
```

Παράδειγμα 1.1. *Να γραφεί κατάλληλη συνάρτηση του MATLAB η οποία να δέχεται σαν όρισμα τους συντελεστές α, β, γ του τριωνύμου $\alpha\, x^2 + \beta\, x + \gamma = 0$ και να επιστρέφει τις ρίζες του τριωνύμου, αν υπάρχουν.*

Γράφουμε τον εξής κώδικα για την λύση πρωτοβάθμιας εξίσωσης και τον σώζουμε με το όνομα αρχείου solve1.m:

```
function r=solve1(a, b)
%Solve equation: ax+b=0
%Call: r=solve1(a,b)
if a~=0
        r=-b/a;
```

```
    elseif b~=0
        r=NaN;
        disp('no solution')
       else r=NaN;
                 disp('undefined solution')
end
```

Κατόπιν γράφουμε τον ακόλουθο κώδικα και τον σώζουμε με το όνομα αρχείου solve2.m:

```
function [r1,r2,det] = solve2(a,b,c)
%Solution of a polynomial of first or second degree
%Calls:  r = solve2(a,b), (ax+b=0)
%      [r1,r2] = solve2(a,b,c), (ax^2+bx+c=0)
%      [r1,r2,det] = solve2(a,b,c), det=determinant
if nargin==2
        r1=solve1(a,b);
    %1st degree, number of input parameters=2
else
if a==0
    %1st degree
        disp('1st degree')
        r1=solve1(b,c);
        r2=NaN;
        det=NaN;
    else
    %2nd degree, number of input parameters=3
       d=b^2-4*a*c;
       if d==0
              r1=-b/(2*a);
                  r2=r1;
                  disp('double root');
       else
          r1=(-b+sqrt(d))/(2*a);
          r2=(-b-sqrt(d))/(2*a);
       end;
    %if output parameters = 3,then export also the determinant
       if nargout==3
              det=d;
         end
    end
end
```

Στο παράθυρο εντολών γράφουμε διάφορες περιπτώσεις τριωνύμων και παίρνουμε τις απαντήσεις:

```
>> r=solve2(3,12)
r =
    -4
>> r=solve2(0,12)
no solution
r =
   NaN
>> r=solve2(0,0)
undefined solution
r =
   NaN
>> [r1,r2]=solve2(1,-5,6)
r1 =
     3
r2 =
     2
>> [r1,r2,det]=solve2(1,-5,6)
r1 =
     3
r2 =
     2
det =
     1
>> [r1,r2]=solve2(0,-5,6)
1st degree
r1 =
    1.200000000000000e+000
r2 =
   NaN
>> [r1,r2,det]=solve2(0,-5,6)
1st degree
r1 =
    1.200000000000000e+000
r2 =
   NaN
det =
   NaN
>> [r1,r2,det]=solve2(0,0,6)
1st degree
no solution
r1 =
   NaN
r2 =
```

```
   NaN
det =
   NaN
>> [r1,r2,det]=solve2(0,0,0)
1st degree
undefined solution
r1 =
   NaN
r2 =
   NaN
det =
   NaN
```

Βλεπουμε ότι για οποιονδήποτε συνδυασμό τιμών δίνει την σωστή απάντηση και όταν δεν υφίσταται κάποια μεταβλητή εξόδου τότε τής δίνει την τιμή NaN=Not a Number. Επίσης έχουμε ήδη δημιουργήσει βοήθεια για την συνάρτησή μας:

```
>> help solve2
 Solution of a polynomial of first or second degree
 Calls:  r = solve2(a,b), (ax+b=0)
       [r1,r2] = solve2(a,b,c), (ax^2+bx+c=0)
       [r1,r2,det] = solve2(a,b,c), det=determinant
```

Η εντολή nargin = number of arguments input δίνει τον αριθμό των ορισμάτων εισόδου, ώστε να καταλαβαίνει η συνάρτηση εάν έχουμε πολυώνυμο πρώτου ή δευτέρου βαθμού.
Η εντολή nargout = number of arguments output δίνει τον αριθμό των ορισμάτων εξόδου, ώστε να καταλαβαίνει η συνάρτηση εάν θέλουμε μόνον τις δύο ρίζες ή εάν θέλουμε και την διακρίνουσα του τριωνύμου.
Οτιδήποτε αρχίζει με % θεωρείται σχόλιο.
Είναι πολύ σημαντικό να τονίσουμε ότι η οποιαδήποτε επώνυμη συνάρτηση θα εκτελεστεί μόνον όταν ο τρέχων φάκελλος εργασίας (Current Folder) περιέχει αυτήν την συνάρτηση.

1.1.2 Ανώνυμες συναρτήσεις

Στις τελευταίες εκδόσεις του MATLAB μπορούμε να ορίσουμε μία ανώνυμη συνάρτηση, με την έννοια ότι δεν θα δημιουργήσουμε κάποιο αντίστοιχο m-file, οποτεδήποτε την χρειαστούμε και την καλούμε από οπουδήποτε.

Παράδειγμα 1.2. *Να γραφεί κατάλληλη ανώνυμη συνάρτηση του MATLAB η οποία να δέχεται σαν όρισμα τους συντελεστές α, β, γ του τριωνύμου $\alpha\, x^2+\beta\, x+\gamma = 0$ και να επιστρέφει τις ρίζες $\rho_{1,2} = \frac{-\beta \pm \sqrt{\Delta}}{2\,\alpha}$.*

Ορίζουμε στο παράθυρο εντολών την εξής συνάρτηση και έχουμε τα ακόλουθα αποτελέσματα από την κλήση της:

```
>> sol2=@(a,b,c)[(-b+sqrt(b^2-4*a*c))/(2*a),(-b-sqrt(b^2-4*a*c))/(2*a)]
sol2 =
    @(a,b,c)[(-b+sqrt(b^2-4*a*c))/(2*a),(-b-sqrt(b^2-4*a*c))/(2*a)]
>> sol2(1,-5,6)
ans =
     3     2
>> sol2(1,-2,1)
ans =
     1     1
>> sol2(0,0,1)
ans =
   NaN   NaN
>> sol2(0,0,0)
ans =
   NaN   NaN
```

Προσέξτε πόσο λιγότερες γραμμές κώδικα χρειάστηκε να γράψουμε για να επιτύχουμε ουσιαστικά τα ίδια αποτελέσματα με τον ογκωδέστατο κώδικα του Παραδείγματος 1. Γι' αυτόν τον λόγο προτιμάμε πάντοτε τις ανώνυμες συναρτήσεις για κάτι σχετικά απλό.

Είναι αυτονόητο ότι δεν μπορούμε να έχουμε πολλές επιλογές όταν ορίζουμε μία ανώνυμη συνάρτηση, γι' αυτό και δεν μπορούμε να τις χρησιμοποιήσουμε για προχωρημένες προγραμματιστικά εργασίες. Μπορούμε όμως άνετα να τις χρησιμοποιήσουμε για οποιονδήποτε απλό ορισμό μαθηματικής συνάρτησης χρειαστούμε.

1.1.3 Γραφικές παραστάσεις

Μπορούμε να ορίσουμε ανώνυμες συναρτήσεις και να κάνουμε την γραφική τους παράσταση με το MATLAB, με την απαραίτητη προϋπόθεση ότι ορίζουμε τις πράξεις διανυσματικά, δηλ. με χρήση της τελίτσας. Έστω η συνάρτηση:

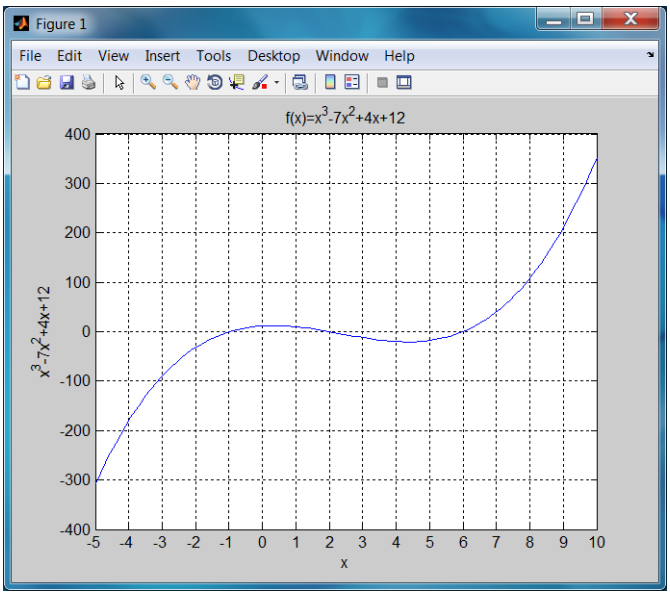
$$f(x) = x^3 - 7\,x^2 + 4\,x + 12 \tag{1}$$

Γράφουμε τις εντολές:

```
>> f=@(x)x.^3-7*x.^2+4*x+12
f =
    @(x)x.^3-7*x.^2+4*x+12
>> x=linspace(-5,10);
>> plot(x,f(x),'-')
set(gca,'XTick',-5:10)
```

```
xlabel('x')
ylabel('{x}^{3}-7{x}^{2}+4x+12')
title('f(x)={x}^{3}-7{x}^{2}+4x+12')
grid('on')
```

Η γραφική παράσταση που προκύπτει είναι:



Σχήμα 1: Γράφημα της ανώνυμης συνάρτησης 1 με το MATLAB

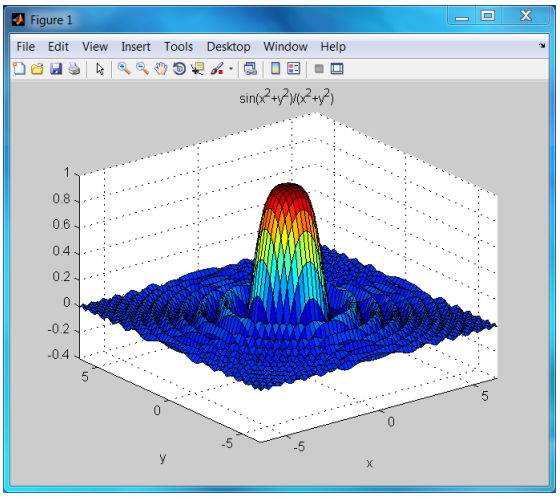
Μπορούμε να κάνουμε γραφική παράσταση συνάρτησης δύο μεταβλητών, όπως η ακόλουθη:

$$f(x,y) = \frac{\sin\left(x^2+y^2\right)}{x^2+y^2} \tag{2}$$

Γράφουμε τις εντολές:

```
>> f=@(x,y)sin(x.^2+y.^2)./(x.^2+y.^2)

f =

    @(x,y)sin(x.^2+y.^2)./(x.^2+y.^2)

>> ezsurf(f)
```

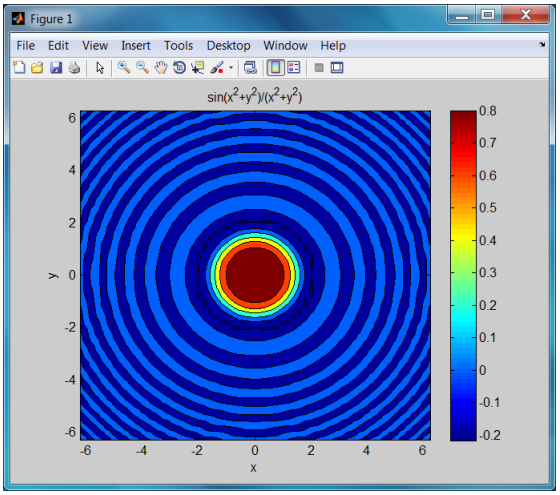
και προκύπτει η γραφική παράσταση:



Σχήμα 2: Γράφημα της ανώνυμης συνάρτησης 2 με το MATLAB

Για την ίδια συνάρτηση μπορούμε να κάνουμε γράφημα ισοϋψών καμπυλών στο xy επίπεδο:

```
>> ezcontourf(f),colorbar
```



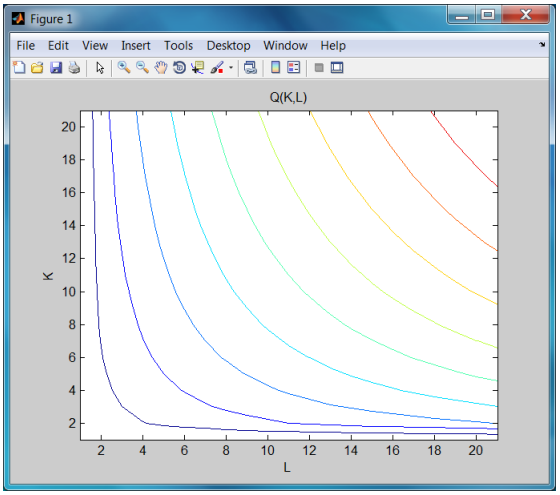
Σχήμα 3: Γράφημα ισοϋψών καμπυλών στο xy επίπεδο της 2 με το MATLAB

Έστω η συνάρτηση παραγωγής Cobb-Douglas:

$$Q(K, L) = 100K^{\frac{3}{5}}L^{\frac{2}{5}} \tag{3}$$

Θα κάνουμε γραφική παράσταση των ισοϋψών καμπυλών αυτής (των καμπυλών ισο-παραγωγής):

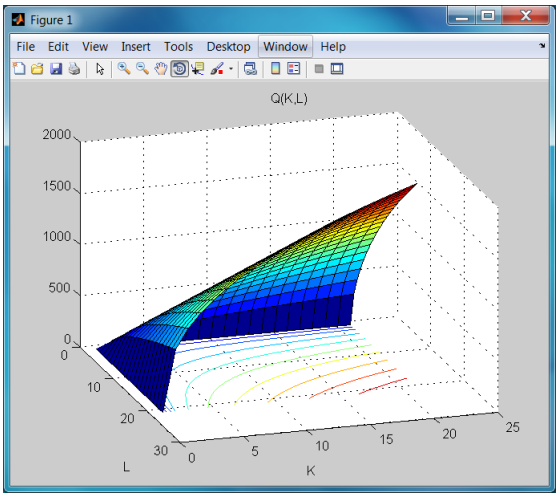
```
>> Q=@(K,L)100*K.^(3/5).*L.^(2/5)

Q =

    @(K,L)100*K.^(3/5).*L.^(2/5)

>> [K,L] = meshgrid(0:1:20,0:1:20);
>> Qp=Q(K,L);
>> contour(Qp)
>> xlabel('L'),ylabel('K'),title('Q(K,L)')
```



Σχήμα 4: Γράφημα καμπυλών ισο-παραγωγής της Cobb-Douglas 3 με το MATLAB

Επίσης μπορούμε να έχουμε την γραφική παράσταση της συνάρτησης παραγωγής και τις ισοϋψείς καμπύλες της ταυτόχρονα με τις επιπλέον εντολές:

```
>> close
>> surfc(Qp)
>> xlabel('L'),ylabel('K'),title('Q(K,L)')
```



Σχήμα 5: Συνδυασμένο γράφημα της Cobb-Douglas 3 με το MATLAB

1.2 M-files: η καρδιά του MATLAB

Πίσω από οποιαδήποτε εργασία, αποτέλεσμα ή γραφικό του MATLAB βρίσκεται, έστω κι αν δεν φαίνεται άμεσα, ένα m-file, δηλ. ένα αρχείο με εντολές που αντιλαμβάνεται το MATLAB. Μπορούμε να πούμε ότι το σύνολο αυτών των εντολών συνθέτουν κατά κάποιο τρόπο μία γλώσσα προγραμματισμού η οποία έχει πολλά κοινά στοιχεία με την FORTRAN και την C. Τις βασικές προγραμματιστικές εντολές αυτής της γλώσσας θα δούμε συνοπτικά ακολούθως.

1.2.1 Εντολές συγκρίσεων

Όλες οι μαθηματικές ισότητες, ανισότητες και αρνήσεις αυτών μπορούν να γραφούν με σύμβολα παρόμοια με τα μαθηματικά σύμβολα, όπως φαίνεται στον Πίνακα 1 .

Μαθηματικό σύμβολο	MATLAB
$=$	==
$\neq$	~=
$>$	>
$\geq$	>=
$\leq$	<=

Πίνακας 1: Σύμβολα μαθηματικών συγκρίσεων στο MATLAB.

Εάν γράψουμε στο παράθυρο εντολών λ.χ. μία σύγκριση αριθμών το MATLAB θα 'απαντήσει' με 1 εάν είναι σωστή και με 0 εάν είναι λάθος αυτό που γράψαμε:

```
>> 5==5
ans =
     1
>> 5==6
ans =
     0
>> 5>4
ans =
     1
>> 5>7
ans =
     0
>> 8~=9
ans =
     1
```

Το ίδιο μπορεί να κάνει και για δύο πίνακες, δηλ. εξετάζει στοιχείο προς στοιχείο εάν ισχύει η σύγκριση που βάλαμε και βγάζει σαν αποτέλεσμα 1(ισχύει) η 0(δεν ισχύει):

```
>> A=[1,2,3;6,5,4;9,8,7]
A =
     1     2     3
     6     5     4
     9     8     7
>> B=[1,-2,3;-6,5,-4;9,-8,-7]
B =
     1    -2     3
    -6     5    -4
     9    -8    -7
>> A==B
ans =
     1     0     1
     0     1     0
     1     0     0
```

1.2.2 Η εντολή for

Η for λέει στο MATLAB να κάνει κάποιες ενέργειες που έχουν να κάνουν με έναν δείκτη, π.χ. i, ο οποίος παίρνει τιμές σε συγκεκριμένο εύρος ακεραίων αριθμών. Ας

υποθέσουμε ότι θέλουμε να υπολογίσουμε στο MATLAB το μερικό άθροισμα:

$$S_n = \sum_{i=1}^{n} \frac{1}{i^2} \tag{4}$$

για $n = 100$. Τότε αρκεί να γράψουμε τον ακόλουθο κώδικα και θα έχουμε το αποτέλεσμα που αναφέρεται κατωτέρω:

```
>> s=0;for i=1:100 s=s+1/i^2;end
>> s
s =
    1.6350
```

Μπορούμε βέβαια να φτιάξουμε μία μικρή συνάρτηση με όρισμα το n και μία ανώνυμη συνάρτηση, $f(i) = \frac{1}{i^2}$ στην προκειμένη περίπτωση, ώστε να υπολογίζουμε τα αθροίσματα για οποιοδήποτε n θελήσουμε:

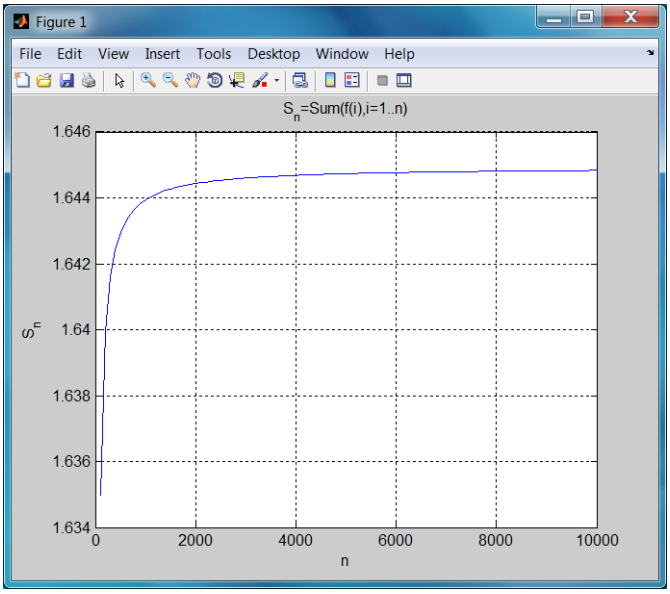
```
function sn=san(n,f)
%Calculates the sum(f(i),i=1..n)
%for a given anonymous function f(n)
%Call: s=san(n,f)
format long
sn=0;
for i=1:n
    sn=sn+f(i);
end
```

Τώρα έχουμε τα εξής για την ανώνυμη συνάρτηση:

```
>> f=@(i)1/i^2
f =
    @(i)1/i^2
>> san(100,f)
ans =
   1.634983900184892
>> san(1000,f)
ans =
   1.643934566681562
>> tic;san(10^6,f),toc
ans =
   1.644933066848770
Elapsed time is 1.193813 seconds.
```

Μπορούμε επίσης να κάνουμε γραφική παράσταση των μερικών αθροισμάτων S_n:

```
nn=[100:100:10000];sn=[];for i=1:size(nn,2) sn=[sn;sanf(nn(i),f)];end
plot(nn,sn)
xlabel('n')
ylabel('S_{n}')
title('S_{n}=Sum(f(i),i=1..n)')
grid('on')
```



Σχήμα 6: Γράφημα των μερικών αθροισμάτων της 4 με το MATLAB

Βλέπουμε ότι όσο αυξάνουμε τον αριθμό των προσθετέων n στο άθροισμα 4 τόσο η τιμή του αθροίσματος δείχνει να συγκλίνει σε μία συγκεκριμένη τιμή. Αυτό είναι αναμενόμενο για την συνάρτηση $\zeta(2)$ του Riemann για την οποία γνωρίζουμε[1] ότι έχει την τιμή:

$$\sum_{i=1}^{\infty} \frac{1}{i^2} = \zeta(2) = \frac{\pi^2}{6} = 1.644934066848226 \tag{5}$$

Με αυτόν τον τρόπο άραγε έχουμε βρει μία μέθοδο για να εξετάζουμε εάν μία σειρά συγκλίνει; Μπορείτε να βρείτε μία σειρά που ακόμα και με το MATLAB να μην είναι δυνατόν να δούμε αν συγκλίνει; Εάν ναι, τότε επικοινωνήστε με τον γράφοντα[2] τις σημειώσεις αυτές.

[1] Βλέπετε λ.χ. εδώ.

[2] Δημήτριος Θ. Χριστόπουλος, dchristop@econ.uoa.gr

1.2.3 Η εντολή if

Είναι μία από τις βασικότερες εντολές γιατί μας δίνει τη δυνατότητα να ελέγχουμε κάθε φορά αν έχει επιτευχθεί ο στόχος μας ή απλά μας δίνει τη δυνατότητα να κάνουμε πολλά πράγματα υπό συνθήκην. Η γενική μορφή της εντολής if είναι:

```
if expression
  statement1
  statement2
  ...
elseif expression
  statement1
  statement2
  ...
else
  statement1
  statement2
  ...
end
```

Εξαρτάται κάθε φορά από την πολυπλοκότητα του προβλήματος το πόσα elseif θα χρησιμοποιήσουμε. Ας υποθέσουμε ότι θέλουμε να υπολογίσουμε το γινόμενο:

$$\prod_{i=1,i\neq 3}^{10} (x_i - x_3)^2 \tag{6}$$

Αρκεί γι' αυτόν τον σκοπό η χρήση ενός απλού if μέσα σε ένα απλό for:

```
>> p=1;
for i=1:10
    if i~= 3
        p=p*(x(i)-x(3))^2;
    end
end
>> p
p =
10160640000000000000000000
```

1.2.4 Η εντολή while

Όταν δεν γνωρίζουμε πόσα βήματα θα απαιτηθούν για την λύση ενός προβλήματος, τότε χρησιμοποιούμε την εντολή while. Η γενική μορφή της εντολής είναι:

```
while expression
```

```
        statement1
        statement2
        ...
      end
```

Ας υποθέσουμε ότι θέλουμε να δούμε για ποια τιμή του n το $n!$ γίνεται μεγαλύτερο από 10^{100}. Τότε αρκεί μία εντολή while και ελάχιστος κώδικας:

```
>> p=10^100;i=1;while factorial(i)<p i=i+1; end
>> i
i =
    70
>> factorial(i-1)-p,factorial(i)-p
ans =
   -9.828877547571859e+099
ans =
    1.978571669969890e+099
```

όπου κάναμε και την επαλήθευσή μας.
Σημαντική παρατήρηση: μπορούμε πάντα να διακόψουμε την εκτέλεση του προγράμματος πληκτρολογώντας CONTROL + C.

1.2.5 Η εντολή switch

Η εντολή αυτή είναι θυμίζει αρκετά την εντολή GO TO της γλώσσας προγραμματισμού FORTRTAN, απλά δέχεται σαν όρισμα όχι μόνον αριθμούς όπως η εντολή GO TO, αλλά και μεταβλητές χαρακτήρων. Η γενική μορφή της εντολής είναι:

```
switch  key
          case {for execution if key=key_1}
            statement1
            statement2
            ...
          case {key_1, key_2, key_3,...}
            statement1
            statement2
            ...
          otherwise,
            statement1
            statement2
            ...
end
```

Σν παράδειγμα θα δημιουργήσουμε έναν μετατροπέα θερμοκρασιών από βαθμούς Celsius, Fahrenheit, Kelvin σε οποιαδήποτε από τις τρεις κλίμακες. Θα δίνουμε μία

θερμοκρασία και θα μας την μετατρέπει στις υπόλοιπες δύο κλίμακες θερμοκρασιών. Αποθηκεύουμε τον κώδικα με το όνομα alltemps.m.

```
format short
T=input('Give the temperature:  ');
deg=input('Give scale in quotes: C(Celsius), F(Fahrenheit), K(Kelvin):  ')
switch deg
    case {'C'}
        F=9/5*T+32;
        disp('In Fahrenheit scale (F) it is:')
        disp(F)
        disp(' ')
        K=T+273.15;
        disp('In Kelvin scale (K) it is:')
        disp(K)
    case {'F'}
        C=(T-32)*5/9;
        disp('In Celsius scale (C) it is:')
        disp(C)
        disp(' ')
        K=(T+459.67)*5/9;
        disp('In Kelvin scale (K) it is:')
        disp(K)
    case {'K'}
        C=T-273.15;
        disp('In Celsius scale (C) it is:')
        disp(C)
        disp(' ')
        F=T*9/5-459.67;
        disp('In Fahrenheit scale (F) it is:')
        disp(F)
    otherwise
        disp(' ')
        disp(['unknown scale:  ' deg])
end
```

Τρέχουμε τώρα τον κώδικα για την μηδενική θερμοκρασία και έχουμε:

```
>> alltemps
Give the temperature:  0
Give scale (in quotes): C(Celsius),F(Fahrenheit),K(Kelvin): 'C'
deg =
C
In Fahrenheit scale (F) it is:
```

```
    32

In Kelvin scale (K) it is:
  273.1500
>> alltemps
Give the temperature:  0
Give scale in quotes: C(Celsius), F(Fahrenheit), K(Kelvin):  'F'
deg =
F
In Celsius scale (C) it is:
  -17.7778

In Kelvin scale (K) it is:
  255.3722
>> alltemps
Give the temperature:  0
Give scale in quotes: C(Celsius), F(Fahrenheit), K(Kelvin):  'K'
deg =
K
In Celsius scale (C) it is:
 -273.1500

In Fahrenheit scale (F) it is:
 -459.6700
>> alltemps
Give the temperature:  0
Give scale in quotes: C(Celsius), F(Fahrenheit), K(Kelvin):  'R'
deg =
R

unknown scale:  R
```

Δεν ενσωματώσαμε την κλίμακα θερμοκρασιών Rankine.

1.3 Ασκήσεις

1. Να δημιουργήσετε με χρήση της εντολής for τους πίνακες και να τους εμφανίσετε όπως παρατίθενται ακολούθως:

$$A = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 2 & 1 & \frac{2}{3} & \frac{1}{2} & \frac{2}{5} \\ 3 & \frac{3}{2} & 1 & \frac{3}{4} & \frac{3}{5} \\ 4 & 2 & \frac{4}{3} & 1 & \frac{4}{5} \\ 5 & \frac{5}{2} & \frac{5}{3} & \frac{5}{4} & 1 \end{pmatrix}, B = \begin{pmatrix} \frac{1}{3} & \frac{1}{5} & \frac{1}{7} & \frac{1}{9} & \frac{1}{11} \\ \frac{2}{5} & \frac{2}{7} & \frac{2}{9} & \frac{2}{11} & \frac{2}{13} \\ \frac{3}{7} & \frac{1}{3} & \frac{3}{11} & \frac{3}{13} & \frac{1}{5} \\ \frac{4}{9} & \frac{4}{11} & \frac{4}{13} & \frac{4}{15} & \frac{4}{17} \\ \frac{5}{11} & \frac{5}{13} & \frac{1}{3} & \frac{5}{17} & \frac{5}{19} \end{pmatrix}$$

2. Να υπολογίσετε με την εντολή for το άθροισμα:

$$\sum_{n=1}^{20} (-1)^n \frac{x^n}{n!}$$

για την τιμή $x = 5$. Μπορείτε να αναγνωρίσετε την συνάρτηση που κρύβεται πίσω από το άθροισμα αυτό ; Ποιό είναι το απόλυτο σφάλμα της προσέγγισης που βρήκαμε ;

3. Να βρείτε πόσους όρους N πρέπει να πάρουμε ώστε το ακόλουθο άθροισμα:

$$\sum_{k=1}^{N} \frac{1}{k}$$

να γίνει μεγαλύτερο από 10.

4. Να δημιουργήσετε κώδικα στο MATLAB ο οποίος όταν πληκτρολογρείτε το όνομά του στο παράθυρο εντολών να κάνει τα ακόλουθα:

   (α′) Να διαβάζει μία απόσταση σε χιλιόμετρα ή μίλια ή ναυτικά μίλια

   (β′) Να μετατρέπει την απόσταση στις άλλες δύο μονάδες και να εμφανίζει τα αποτελέσματα στην οθόνη.