«Αδαής: το πρόσωπο που αγνοεί ορισμένες από τις γνώσεις που κατέχεις καλά και διαθέτει κάποιες άλλες, τις οποίες αγνοείς παντελώς».

Ambrose Bierce, *The Devil's Dictionary*, 1911

15 MATLAB ΓΙΑ ΓΕΝΙΚΗ ΧΡΗΣΗ

Πέρα από τις πολλαπλές δυνατότητες υπολογισμών, το MATLAB παρέχει και μία πληθώρα από εντολές και συναρτήσεις, που είναι απαραίτητες σε πολλά από τα γνωστικά αντικείμενα των σπουδών Μηχανικού. Κάποιες από αυτές παρουσιάζονται παρακάτω.

15.1 ΕΠΙΛΥΣΗ ΓΡΑΜΜΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

15.1.1 ΕΙΣΑΓΩΓΗ

Έστω το 3x3 σύστημα γραμμικών εξισώσεων

$$\begin{aligned} 3x_1 + 2x_2 - x_3 &= 10 \\ -x_1 + 3x_2 + 2x_3 &= 5 \\ x_1 - x_2 - x_3 &= -1 \end{aligned} \qquad (15.1)$$

Το σύστημα αυτό μπορεί να γραφεί στη μορφή

$$A \cdot x = b \qquad (15.2)$$

όπου

$$A = \begin{pmatrix} 3 & 2 & -1 \\ -1 & 3 & 2 \\ 1 & -1 & -1 \end{pmatrix},\ x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}\ \kappa\alpha\iota\ \ b = \begin{pmatrix} 10 \\ 5 \\ -1 \end{pmatrix} \qquad (15.3)$$

Ισοδύναμα μπορούμε να το γράψουμε στη μορφή:

$$x^T \cdot A^T = b^T \qquad (15.4)$$

όπου με ^{T} συμβολίζουμε τον ανάστροφο πίνακα και οι A, x, b δίνονται και πάλι από τις σχέσεις (15.3).

Το σύστημα αυτό έχει μοναδική λύση όταν ο πίνακας A είναι μη-ιδεάζων (non singular), δηλαδή η τάξη του είναι ίση με τον αριθμό των αγνώστων (τρεις στην προκειμένη περίπτωση) ή ισοδύναμα η ορίζουσά του δεν είναι ίση με το μηδέν[29]. Σε διαφορετική περίπτωση το σύστημα είτε δεν έχει λύση, είτε έχει άπειρες λύσεις. Παρακάτω εισάγουμε στο MATLAB τους πίνακες (15.3) και ελέγχουμε αν πράγματι ο πίνακας είναι μη-ιδεάζων.

```
>> A=[3,2,-1;-1,3,2;1,-1,-1]      % ορίζουμε τον A
A =
     3     2    -1
    -1     3     2
     1    -1    -1

>> b=[10;5;-1]                    % ...και τον b
b =
    10
     5
    -1

>> rank(A)                        % η τάξη του A είναι 3
ans =
     3

>> det(A)                         % η ορίζουσα του 1≠0,
ans =                              % άρα θα έχουμε μοναδική
λύση
     1
```

[29] Αυτό μεθερμηνεύεται στο γεγονός πως κάθε γραμμή του πίνακα αντιστοιχεί σε διάνυσμα που δεν προκύπτει ως γραμμικός συνδυασμός άλλων αντίστοιχων διανυσμάτων του συστήματος, είναι δηλαδή γραμμικά ανεξάρτητο από (και άρα όχι παράλληλο με) όλα τα υπόλοιπα.

15.1.2 Λύση γραμμικού συστηματος

Πολλαπλασιάζοντας την εξίσωση (15.2) από αριστερά με τον A^{-1} θα έχουμε

$$A^{-1} \cdot A \cdot x = A^{-1} \cdot b \xrightarrow{A^{-1} \cdot A = I} x = A^{-1} \cdot b$$

Επομένως η λύση του συστήματος είναι η

$$x = A^{-1} \cdot b \qquad (15.5)$$

Ο υπολογισμός στο MATLAB μπορεί να γίνει πολύ απλά ως ακολούθως

```
>> x=inv(A)*b
x =
   -2.0000
    5.0000
   -6.0000
```

Σε περίπτωση που το σύστημα περιγράφονταν από την εξίσωση (5.4), θα πολλαπλασιάζαμε με τον $(A^T)^{-1}$ από δεξιά και θα καταλήγαμε στη λύση

$$x = b^T \cdot (A^T)^{-1} \qquad (15.6)$$

Στο παράδειγμα που ακολουθεί φαίνεται πως καταλήγουμε στο ίδιο αποτέλεσμα, με τη διαφορά ότι το x είναι διάνυσμα – γραμμή και όχι διάνυσμα – στήλη όπως πριν.

```
>> x=b'*inv(A')
x =
   -2.0000    5.0000   -6.0000
```

15.1.3 Επιλυση με διαιρεση πινακων

Το MATLAB μας δίνει επιπλέον τη δυνατότητα να επιλύουμε γραμμικά συστήματα διαιρώντας απευθείας τους πίνακες. Έτσι, για τους πίνακες που έχουν οριστεί από τις σχέσεις (15.3), θα έχουμε:

```
>> x=A\b
x =
   -2.0000
    5.0000
   -6.0000
```

Πρέπει να επισημανθεί ότι εδώ χρησιμοποιούμε τον τελεστή αριστερής διαίρεσης «\» (backslash) και όχι τον γνωστό τελεστή της διαίρεσης «/» (slash). Σε περίπτωση που το σύστημα ορίζονταν από τη σχέση (15.4) τότε η λύση θα προέκυπτε από την κανονική (δεξιά) διαίρεση:

```
x=b'/A'
x =
   -2.0000    5.0000   -6.0000
```

Εδώ και πάλι παρατηρούμε ότι το αποτέλεσμα είναι διάνυσμα – γραμμή. Για να επιβεβαιώσουμε τη λύση που έχουμε βρει, μπορούμε να υπολογίσουμε την παράσταση $\boldsymbol{A\,x - b}$ και να δούμε εάν αυτή ισούται με το μηδέν. Ο υπολογισμός μας θα καταδείξει πως:

```
>> A*x-b
ans =

  1.0e-014 *

         0
    0.1776
         0
```

Από την άλλη, το **eps**[30], είναι

```
>> eps
ans =
  2.2204e-016
```

Αυτό σημαίνει ότι η λύση στην οποία καταλήγουμε είναι, σύμφωνα με το MATLAB, κοντά στην ακρίβεια υπολογισμών.
Σε πολλές περιπτώσεις, έχουμε περισσότερες εξισώσεις από ότι αγνώστους, όπως στο ακόλουθο παράδειγμα:

$$\begin{matrix} x - y = 0 \\ y = 2 \\ x = 1 \end{matrix} \text{ ή } A\,x = b, \text{ όπου } A = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \text{ και } b = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$$

[30] eps = η σχετική ακρίβεια υπολογισμών με αριθμούς κινητής υποδιαστολής.

Σε αυτές τις περιπτώσεις, το σύστημα λέγεται υπερπροσδιορισμένο (overdetermined), και δεν έχει λύση. Παρόλα αυτά, το MATLAB μπορεί να μας οδηγήσει σε μία προσεγγιστική λύση, δηλαδή σε τιμές του διανύσματος x που ελαχιστοποιούν την παράσταση $\boldsymbol{r} = \boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{b}$. Πράγματι:

```
x =
    1.3333
    1.6667
```

Αντικαθιστώντας, βρίσκουμε ότι:

```
>> t=A*x-b
t =
   -0.3333
   -0.3333
    0.3333
```

Στην περίπτωση αυτή το MATLAB εφαρμόζει τη μέθοδο των ελαχίστων τετραγώνων, για να ελαχιστοποιήσει το μέτρο του διανύσματος t, δηλαδή την ποσότητα $t = \sqrt{t^2(1) + t^2(2) + t^2(3)}$

15.2 ΠΟΛΥΩΝΥΜΑ

15.2.1 ΠΑΡΑΣΤΑΣΗ ΠΟΛΥΩΝΥΜΩΝ ΣΤΟ MATLAB

Στο MATLAB τα πολυώνυμα παριστάνονται ως διανύσματα – γραμμές με τους συντελεστές τους να αποτελούν τα στοιχεία του διανύσματος με φθίνουσα σειρά, σύμφωνα με τους εκθέτες των αγνώστων. Έτσι, το πολυώνυμο:

$$x^3 - 2x^2 - 5x + 6$$

θα γράφεται ως το διάνυσμα:

$$p = [1 \quad -2 \quad -5 \quad 6]$$

15.2.2 ΡΙΖΕΣ ΠΟΛΥΩΝΥΜΩΝ

Η εύρεση των ριζών ενός πολυωνύμου γίνεται με χρήση της εντολής *roots(x)*, όπου x είναι το διάνυσμα με τους συντελεστές του πολυωνύμου. Βάσει αυτού,

ο υπολογισμός των ριζών ενός πολυωνύμου όπως το $p(x) = x^3 - 2x^2 - 5x + 6$ θα γίνει ως ακολούθως:

```
>> p=[1 -2 -5 6]
p =
     1    -2    -5     6

>> roots(p)
ans =
   -2.0000
    3.0000
    1.0000
```

Επομένως, οι ρίζες είναι οι: $\rho_1 = -2$, $\rho_2 = 3$, $\rho_3 = 1$. Αντίστροφα, με την εντολή *poly()* μας δίνεται η δυνατότητα να κατασκευάσουμε ένα πολυώνυμο, αν είναι γνωστές οι ρίζες του. Οπότε, αν γνωρίζουμε ότι οι ρίζες ενός πολυωνύμου είναι οι $\rho_1 = -2$, $\rho_2 = 3$, $\rho_3 = 1$, τότε μπορούμε να το δημιουργήσουμε, δηλαδή να βρούμε τους συντελεστές των όρων του:

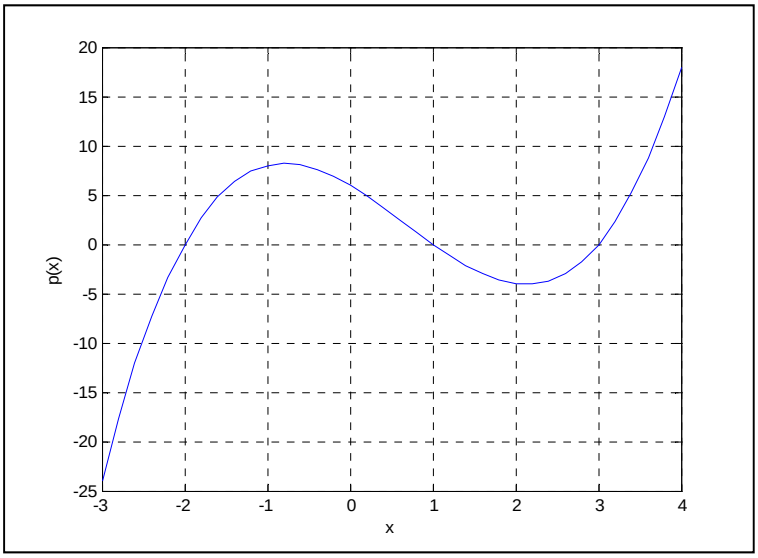
```
>> r=[-2 3 1];
>> p=poly(r)
p =
     1    -2    -5     6
```

15.2.3 ΥΠΟΛΟΓΙΣΜΟΣ ΤΙΜΩΝ ΠΟΛΥΩΝΥΜΟΥ ΚΑΙ ΓΡΑΦΙΚΗ ΠΑΡΑΣΤΑΣΗ

Για τον υπολογισμό τιμών ενός πολυωνύμου χρησιμοποιούμε την εντολή *polyval(p,x)*, όπου p είναι το πολυώνυμο και x είναι το σημείο για το οποίο θέλουμε να υπολογίσουμε την τιμή του. Το x μπορεί να είναι ένα διάνυσμα, οπότε ο υπολογισμός θα γίνει για όλα τα στοιχεία αυτού. Ακολούθως παρουσιάζεται ένα σχετικό παράδειγμα, καθώς και η κατασκευή της γραφικής παράστασης του πολυωνύμου αυτού.

```
>> x=-3:0.2:4;
>> y=polyval(p,x);
>> plot(x,y); grid
>> xlabel('x'); ylabel('p(x)')
```

Η γραφική παράσταση που προέκυψε παρουσιάζεται στο Σχήμα 15.1.



ΣΧΗΜΑ 15.1:ΓΡΑΦΙΚΗ ΠΑΡΑΣΤΑΣΗ ΤΟΥ ΠΟΛΥΩΝΥΜΟΥ P(X).

15.2.4 ΠΡΑΞΕΙΣ ΜΕ ΠΟΛΥΩΝΥΜΑ

Το MATLAB είναι εφοδιασμένο με πολλές εντολές-εγγενείς συναρτήσεις που αφορούν πολυώνυμα. Ο Πίνακας 16.1 περιλαμβάνει τις συχνότερα χρησιμοποιούμενες εντολές με τις οποίες κάνουμε πράξεις μεταξύ πολυωνύμων. Σε κάθε περίπτωση το αποτέλεσμα της πράξης είναι ένα καινούργιο πολυώνυμο, το οποίο μας δίδεται με τη μορφή διανύσματος – γραμμής.

ΠΙΝΑΚΑΣ 15.1 ΟΙ ΒΑΣΙΚΕΣ ΕΝΤΟΛΕΣ ΓΙΑ ΤΗΝ ΕΚΤΕΛΕΣΗ ΠΡΑΞΕΩΝ ΜΕΤΑΞΥ ΠΟΛΥΩΝΥΜΩΝ.

Πράξη	Εντολή	Σχόλια
Πρόσθεση – Αφαίρεση	`C = A ± B`	Τα πολυώνυμα Α, Β πρέπει να έχουν το ίδιο μήκος
Πολλαπλασιασμός πολυωνύμου με αριθμό	`C = λ*A`	όπου λ ένας αριθμός
Πολλαπλασιασμός πολυωνύμων	`C = conv(A,B)`	Υπολογίζει το γινόμενο των δύο πολυωνύμων

Διαίρεση πολυωνύμων	`[q,r] = deconv(A,B)`	Διαιρεί τα πολυώνυμα. Το q είναι το πηλίκο της διαίρεσης και το r το υπόλοιπο

Υπολογισμός παραγώγου	`polyder(A)`
	
	`polyder(A,B)`
	
	`[n d] = polyder(A,B)`
		Υπολογίζει τη παράγωγο του πολυωνύμου Α
		
		Υπολογίζει την παράγωγο του γινομένου **A*B**
		
		Υπολογίζει την παράγωγο του **A/B**, με τη μορφή **n/d**

Ακολουθούν μερικά παραδείγματα πράξεων μεταξύ πολυωνύμων:

```
>> p1=[1 2 -5];          % ορισμός δύο πολυωνύμων
>> p2=[3 -4 2];

>> p1+p2                 % πρόσθεση πολυωνύμων
ans =
     4    -2    -3

>> p1-p2                 % αφαίρεση πολυωνύμων
ans =
    -2     6    -7

>> 3*p1                  % πολλαπλασιασμός με αριθμό
ans =
     3     6   -15

>> conv(p1,p2)           % πολλαπλασιασμός πολυωνύμων
ans =
     3     2   -21    24   -10

>> [q r]=deconv(p2,p1)   % διαίρεση πολυωνύμων
q =
     3
r =
     0   -10    17

>> polyder(p1)           % υπολογισμός παραγώγου
ans =
```

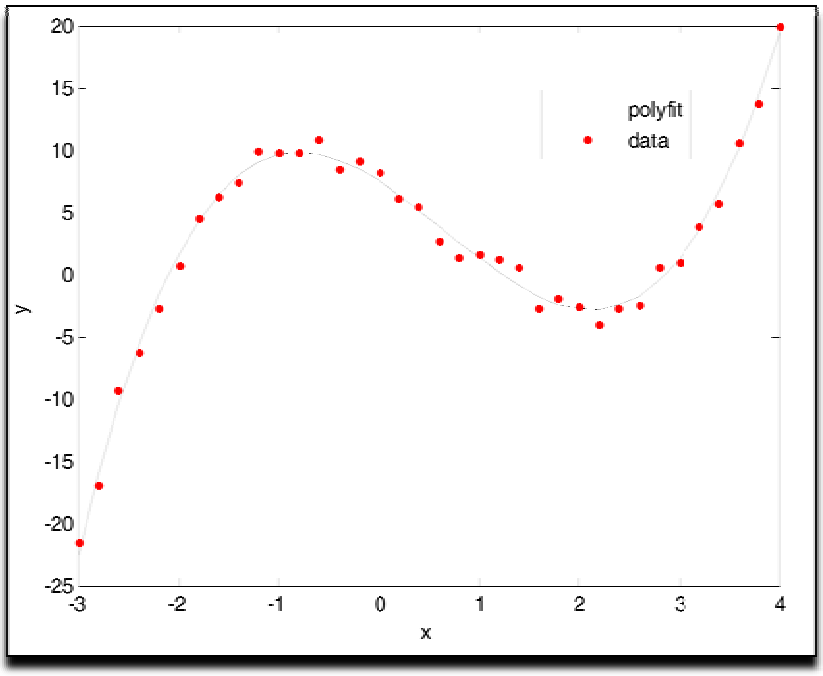
```
     2     2

>> polyder(p1,p2)  % παράγωγος του γινομένου των p1 και p2
ans =
    12     6   -42    24

>> [n d]=polyder(p1,p2)   % παράγωγος του p1/p2
n =
   -10    34   -16
d =
     9   -24    28   -16     4
```

15.2.5 ΠΟΛΥΩΝΥΜΙΚΗ ΠΑΡΕΜΒΟΛΗ

Η πολυωνυμική παρεμβολή είναι μία μέθοδος με την οποία κατασκευάζουμε ένα πολυώνυμο το οποίο προσεγγίζει κατά τον «καλύτερο τρόπο» ένα σύνολο δεδομένων. Με τη χρήση της εντολής *polyfit(x,y,n)* μας δίνεται η δυνατότητα να παρεμβάλουμε ένα πολυώνυμο $\boldsymbol{n}$ βαθμού σε ένα σύνολο δεδομένων (x,y). Έτσι για παράδειγμα, για τα δεδομένα που περιλαμβάνει ο Πίνακας 15.1 και χρησιμοποιώντας την εν λόγω εντολή, θα λάβουμε το πολυώνυμο 3^{ου} βαθμού που οπτικοποιείται στο Σχήμα 15.2.

ΠΙΝΑΚΑΣ 15.1: ΔΕΔΟΜΕΝΑ ΠΑΡΑΔΕΙΓΜΑΤΟΣ ΠΟΛΥΩΝΥΜΙΚΗΣ ΠΑΡΕΜΒΟΛΗΣ.

x	y	x	y	X	y	x	y	x	y
-3	-21.5571	-1.4	7.391	0	8.2612	1.4	0.626	2.8	0.6549
-2.8	-16.9014	-1.2	9.8845	0.2	6.0693	1.6	-2.6343	3	0.9336
-2.6	-9.3082	-1	9.7558	0.4	5.4475	1.8	-1.9415	3.2	3.8736
-2.4	-6.294	-0.8	9.8572	0.6	2.7236	2	-2.5918	3.4	5.6809
-2.2	-2.7382	-0.6	10.8156	0.8	1.3939	2.2	-3.9963	3.6	10.5419
-2	0.7533	-0.4	8.4735	1	1.5924	2.4	-2.6846	3.8	13.7809
-1.8	4.5361	-0.2	9.1836	1.2	1.1855	2.6	-2.4575	4	19.9622
-1.6	6.2039								



ΣΧΗΜΑ 15.2: ΠΑΡΕΜΒΟΛΗ ΕΝΟΣ ΠΟΛΥΩΝΥΜΟΥ 3^{ΟΥ} ΒΑΘΜΟΥ ΣΤΑ ΔΕΔΟΜΕΝΑ ΤΟΥ ΠΙΝ. 15.1.

Οι εντολές MATLAB από τις οποίες προέκυψε το πολυώνυμο αυτό παρουσιάζονται παρακάτω.

```
% Τα δεδομένα x, y είναι φορτωμένα στη μνήμη.

>> p=polyfit(x,y,3)      % παρεμβολή στα δεδομένα (x,y)
p =
    1.0121   -2.0192   -5.1486    7.5174

>> y3=polyval(p,x);   % υπολ. τιμών με καινούργιο πολυώνυμο

>> plot(x,y,'k',x,y3,'r.')   % γραφική απεικόνιση
>> xlabel('x')
>> ylabel('y')
>> legand('polyfit', 'data')
```

Η επιλογή του βαθμού του πολυωνύμου εξαρτάται από τα δεδομένα που έχουμε. Στην προκείμενη περίπτωση διαπιστώσαμε πως το καλύτερο δυνατό αποτέλεσμα προέκυπτε με ένα πολυώνυμο $3^{ου}$ βαθμού. Το κριτήριο επιλογής πολυωνύμου είναι η ελαχιστοποίηση της απόκλισης μεταξύ δεδομένων και καμπύλης. Για το σκοπό αυτό χρησιμοποιείται συνήθως το τετράγωνο της ευκλείδειας απόστασης δεδομένων και καμπύλης ως κριτήριο ελαχιστοποίησης.

15.3 ΥΠΟΛΟΓΙΣΜΟΣ ΟΛΟΚΛΗΡΩΜΑΤΩΝ

Το MATLAB υποστηρίζει τον υπολογισμό ολοκληρωμάτων με μία ποικιλία μεθόδων, που τις υλοποιεί με τη βοήθεια αντίστοιχων εντολών. Έτσι λοιπόν, για τον υπολογισμό απλών ολοκληρωμάτων μπορούμε να χρησιμοποιήσουμε την εντολή:

```
quad('συνάρτηση', κάτω όριο, άνω όριο)
```

η οποία χρησιμοποιεί την μέθοδο ολοκλήρωσης Simpson[31]. Η προς ολοκλήρωση συνάρτηση μπορεί είτε να τοποθετηθεί εντός απλών εισαγωγικών, είτε να βρίσκεται σ' ένα ξεχωριστό αρχείο (m-file), οπότε η σύνταξη είναι:

```
quad(@όνομα_αρχείου, κάτω όριο, άνω όριο)
```

Τα ορίσματα κάτω όριο και άνω όριο είναι το κάτω και άνω όριο ολοκλήρωσης αντιστοίχως. Με τον ίδιο τρόπο συντάσσονται και οι εντολές *quadl* και *trapz* οι οποίες χρησιμοποιούν άλλες μεθόδους αριθμητικής ολοκλήρωσης, καθώς και η *dblquad* που χρησιμοποιείται για τον υπολογισμό διπλών ολοκληρωμάτων.

- **Παράδειγμα: Υπολογισμός μήκους καμπύλης**

Έστω μια καμπύλη στο χώρο η οποία δίνεται με την ακόλουθη παραμετρική μορφή:

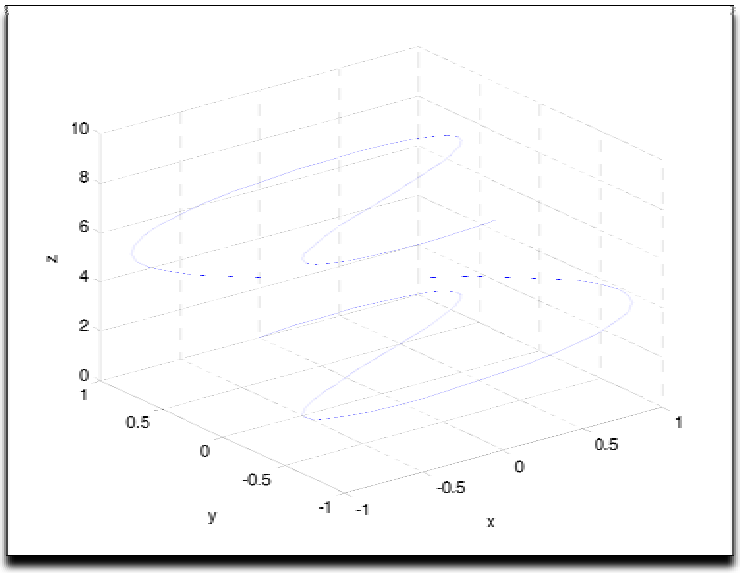
$$s(t): \quad x(t) = \sin(2t) \ , \ y(t) = \cos(t) \ , \ z(t) = t \quad t \, \epsilon \, [0, 3\pi]$$

Η γραφική παράσταση αυτής μπορεί να γίνει με χρήση της εντολής *plot3*, όπως φαίνεται στο Σχήμα 15.3. Οι εντολές με τις οποίες προέκυψε η γραφική παράσταση παρατίθενται ακολούθως.

---

[31] Πρόκειται για μέθοδο αριθμητικής ανάλυσης για τον υπολογισμό Ορισμένων ολοκληρωμάτων.

```
>> t = 0:0.1:3*pi;
>> plot3(sin(2*t),cos(t),t)
>> grid
>> xlabel('x')
>> ylabel('y')
>> zlabel('z')
```



ΣΧΗΜΑ 15.3: ΓΡΑΦΙΚΗ ΠΑΡΑΣΤΑΣΗ ΤΗΣ ΚΑΜΠΥΛΗΣ S(T).

Το μήκος της καμπύλης ορίζεται από την σχέση

_____

και στην προκείμενη περίπτωση θα είναι:

_____

Έτσι σύμφωνα με όσα αναφέρθηκαν παραπάνω ο υπολογισμός γίνεται με τη μέθοδο Simpson, σύμφωνα με την παρακάτω εντολή:

```
>> s=quad('sqrt(4*cos(2*t).^2 + sin(t).^2 + 1)',0,3*pi)
s =
   17.2220
```

Στο ίδιο αποτέλεσμα θα οδηγούμασταν εάν γράφαμε την συνάρτηση σε ένα ξεχωριστό αρχείο (m-file), με το όνομα hcurve.m, όπως φαίνεται παρακάτω:

```
% Περιεχόμενο αρχείου hcurve.m
function f = hcurve(t)
f = sqrt(4*cos(2*t).^2 + sin(t).^2 + 1);
end
```

Κατόπιν, για να υπολογίσουμε το ορισμένο ολοκλήρωμα, θα κάναμε χρήση του ονόματος του αρχείου που περιέχει τη συνάρτηση:

```
>> s = quad(@hcurve,0,3*pi)
s =
   17.2220
```

15.4 ΣΥΜΒΟΛΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ ΚΑΙ ΣΥΝΑΡΤΗΣΕΙΣ

Οι συμβολικές μεταβλητές και συναρτήσεις μας παρέχουν τη δυνατότητα να εκτελούμε πράξεις με το MATLAB, χωρίς να απαιτείται να προσδιορίζουμε τη τιμή που έχει μια μεταβλητή. Έτσι, μπορούμε να εκτελούμε πράξεις μεταξύ αριθμητικών παραστάσεων, να υπολογίζουμε παραγώγους και ολοκληρώματα συναρτήσεων, να επιλύουμε εξισώσεις και συστήματα εξισώσεων με παραμέτρους, διαδικασίες που δεν θα μπορούσαν να υλοποιηθούν με την υπολογιστική προσέγγιση που έχουμε γνωρίσει μέχρι στιγμής.

15.4.1 ΣΥΜΒΟΛΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ, ΣΤΑΘΕΡΕΣ ΚΑΙ ΕΚΦΡΑΣΕΙΣ

Μία μεταβλητή είναι συμβολική όταν δύναται να χρησιμοποιηθεί ως στοιχείο αλγεβρικών πράξεων, χωρίς να απαιτείται η ανάθεση τιμών σε αυτή. Για να οριστεί μία μεταβλητή ως συμβολική χρησιμοποιούμε την εντολή *syms:*

```
>> syms x y
```

Ο ορισμός συμβολικών σταθερών γίνεται με την συνάρτηση *sym()*, όπως φαίνεται στο παράδειγμα που ακολουθεί:

```
>> delta = sym('1/10')
delta =
1/10
```

Έχοντας ορίσει με συμβολικό τρόπο μεταβλητές, μπορούμε να τις συνδυάσουμε και να δημιουργήσουμε συμβολικές εκφράσεις, όπως φαίνεται στο παρακάτω παράδειγμα:

```
>> syms s t A
>> f = s^2 + 4*s + 5
f =
s^2+4*s+5
```

Με τον ίδιο τρόπο μπορούμε να κατασκευάσουμε διανύσματα και πίνακες που να έχουν ως στοιχεία συμβολικές μεταβλητές, όπως επιδεικνύεται ακολούθως:

```
>> syms x
>> A=[1 2; x x^2]
A =
[   1,   2]
[   x, x^2]
```

15.4.2 ΠΡΑΞΕΙΣ ΜΕΤΑΞΥ ΣΥΜΒΟΛΙΚΩΝ ΕΚΦΡΑΣΕΩΝ

- **Πολυώνυμα**

Ορίζοντας δύο πολυώνυμα με συμβολικό τρόπο, μπορούμε να εκτελέσουμε μεταξύ αυτών όλες τις γνωστές πράξεις. Στα παραδείγματα που ακολουθούν φαίνεται πως μπορεί να γίνει κάτι τέτοιο:

```
>> syms x

>> p1=x^2+2*x-5                  % ορισμός πολυωνύμων
p1 =
x^2+2*x-5
>> p2=3*x^2-4*x+2
p2 =
3*x^2-4*x+2

>> p1+p2                         % πρόσθεση πολυωνύμων
ans =
4*x^2-2*x-3

>> p1-p2                         % αφαίρεση πολυωνύμων
ans =
-2*x^2+6*x-7

>> 3*p1                          % πολ/σμός με αριθμό
ans =
3*x^2+6*x-15

>> p1*p2                         % πολ/σμός πολυωνύμων
ans =
(x^2+2*x-5)*(3*x^2-4*x+2)

>> expand(p1*p2)                 % αναπτ. μορφή γινομένου
ans =
3*x^4+2*x^3-21*x^2+24*x-10
```

Επιπλέον, μας δίδεται η δυνατότητα να παραγοντοποιήσουμε ένα πολυώνυμο καθώς και να απλοποιήσουμε μια κλασματική παράσταση με τις εντολές *factor* και *simplify* αντιστοίχως:

```
>> syms x
>> p1=x^3 +2*x^2 +5*x +10;
>> p2=x^2+5;

>> factor(p1)              % παραγοντοποίηση πολυωνύμου
ans =
(x+2)*(x^2+5)

>> H=p1/p2                 % ορισμός κλασματικής παράστασης
H =
```

```
(x^3+2*x^2+5*x+10)/(x^2+5)

>> simplify(H)              % απλοπ. κλασματικής παράστασης
ans =
x+2
```

Τέλος, κάνοντας χρήση των εντολών *sym2poly* και *poly2sym* μπορούμε να μετατρέψουμε ένα συμβολικά ορισμένο πολυώνυμο σε διάνυσμα που περιέχει τους συντελεστές του και αντίστροφα. Στο παράδειγμα που ακολουθεί παρουσιάζεται μια τέτοια μετατροπή.

```
>> syms x
>> p1=x^3 +2*x^2 +5*x +10;

>> pvector=sym2poly(p1)     % μετατρ. συμβολικά ορισμένου
 pvector =                  % πολυωνύμου σε διάνυσμα
     1     2     5    10

>> psym=poly2sym(pvector)   % μετατρ. πολυωνύμου-διανύσματος
psym =                      % σε συμβολικό πολυώνυμο
x^3+2*x^2+5*x+10
```

- **Τριγωνομετρικές εκφράσεις**

Εξαιρετικά χρήσιμος είναι ο συμβολικός μετασχηματισμός τριγωνομετρικών εκφράσεων. Στα παραδείγματα που ακολουθούν παρουσιάζονται ορισμένοι τέτοιοι μετασχηματισμοί.

```
>> syms x y

>> A = sin(x+y);              % ανάπτυγμα αθροίσματος γωνιών
>> A = expand(A)
A =
sin(x)*cos(y)+cos(x)*sin(y)

>> C = sin(x)^2 + cos(x)^2;   % απλοποίηση με χρήση
>> C = simplify(C)            % τριγωνομετρικής ταυτότητας
C =
1
```

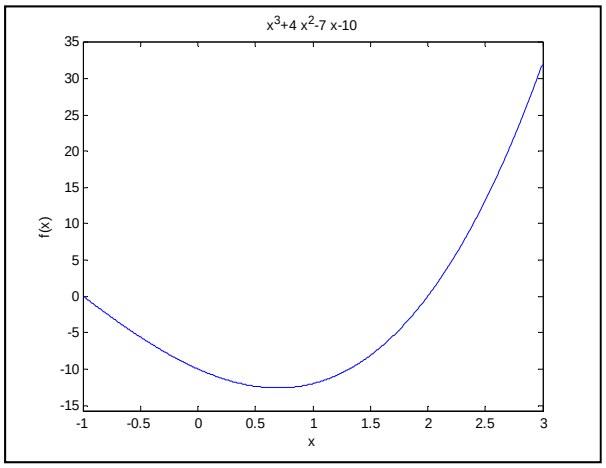
15.4.3 ΥΠΟΛΟΓΙΣΜΟΣ ΤΙΜΩΝ ΚΑΙ ΓΡΑΦΙΚΗ ΑΠΕΙΚΟΝΙΣΗ ΣΥΜΒΟΛΙΚΩΝ ΕΚΦΡΑΣΕΩΝ

Σε πολλές περιπτώσεις θα θέλαμε να υπολογίσουμε την τιμή ή να κατασκευάσουμε τη γραφική παράσταση μίας συμβολικής έκφρασης. Αυτό μπορεί να γίνει με τις εντολές *subs* και *ezplot* αντίστοιχα, όπως εξηγείται και επιδεικνύεται στα παραδείγματα που ακολουθούν.

```
>> syms x                              % ορισμός συμβ. έκφρασης
>> E = x^3 -14*x^2 +65*x -100;

>> F = subs(E,x,7.1)                   % υπολογισμός της τιμής
                                       % της F στο x=7.1
F =
13.671
```

```
>> syms x                              % ορισμός συμβ. έκφρασης
>> f=x^3 + 4*x^2 - 7*x - 10;
>> ezplot(f,-1,3)                      % κατασκευή γραφ. παράστ.
>> ylabel('f(x)')                      % προσθήκη λεζάντας
                                       % στον άξονα των y
```

Η γραφική παράσταση που προκύπτει από τις τελευταίες εντολές που παραδείγματος παρουσιάζεται στο Σχήμα 15.4.



ΣΧΗΜΑ 15.4: ΓΡΑΦΙΚΗ ΠΑΡΑΣΤΑΣΗ ΤΗΣ ΣΥΜΒΟΛΙΚΑ ΟΡΙΣΜΕΝΗΣ ΣΥΝΑΡΤΗΣΗΣ F(X).

15.4.4 ΕΠΙΛΥΣΗ ΑΛΓΕΒΡΙΚΩΝ ΕΞΙΣΩΣΕΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ

Με την εντολή *solve* μπορούμε να επιλύσουμε τόσο συμβολικές αλγεβρικές εξισώσεις, όσο και συστήματα αλγεβρικών εξισώσεων που περιέχουν συμβολικές μεταβλητές. Στο παράδειγμα που ακολουθεί επιδεικνύεται ο τρόπος με τον οποίο μπορούμε να επιλύσουμε μια αλγεβρική εξίσωση[32].

```
>> syms x
>> equation=x^3 -2*x^2 -3*x + 10;

>> rootsOfEquation=solve(equation,x)
rootsOfEquation =
  -2
 2+i
 2-i
```

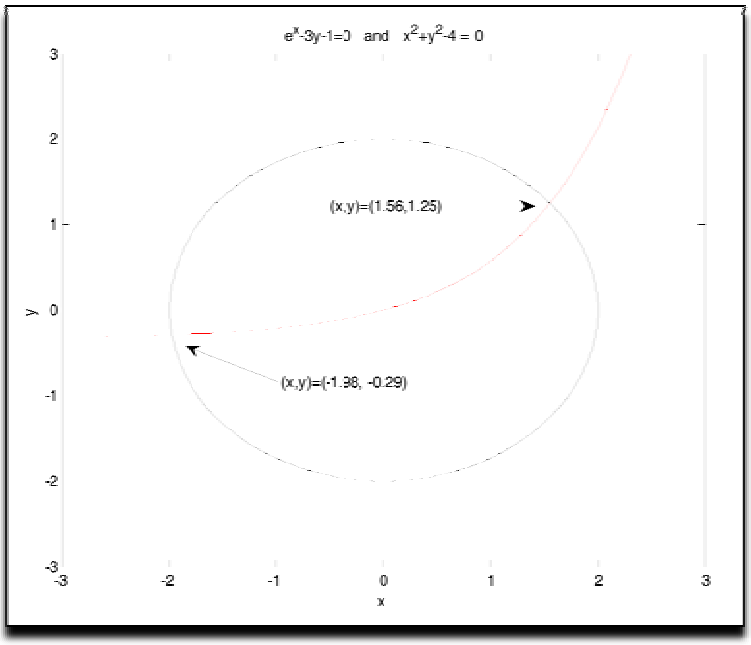
Για να επιλύσουμε ένα σύστημα εξισώσεων χρησιμοποιούμε και πάλι την εντολή *solve* με διαφορετικό όμως τρόπο: η λύση ανατίθεται σε ένα διάνυσμα, με τόσα στοιχεία, όσες και οι ρίζες του συστήματος εξισώσεων. Έτσι, στο παρακάτω παράδειγμα, οι λύσεις ανατίθενται στο διάνυσμα [x1 y1]. Στο παράδειγμα αυτό χρησιμοποιείται επίσης η εντολή *double*, η οποία μετατρέπει αριθμητικές ποσότητες σε διπλής ακριβείας.

```
>> syms x y
>> equation1=exp(x)-3*y-1;  % εξισώσεις προς επίλυση
>> equation2=x^2+y^2-4;     % εξισώσεις προς επίλυση

>> [x1 y1]=solve(equation1,equation2,x,y) ;

>> x1
 x1 =
 1.559512193572005786235870052 5512-0.*i

>> double(x1)
ans =
    1.5595

>> y1
```

[32] Υπενθυμίζεται ότι στην περίπτωση πολ/κών εξισώσεων, μία εξίσωση k βαθμού έχει k ρίζες.

```
y1 =
1.252166809215222232992324479151 9-0.*i

>> double(y1)
ans =
    1.2522
```

Στο παραπάνω παράδειγμα καταλήξαμε στη λύση , ωστόσο κάνοντας τη γραφική παράσταση των δύο συναρτήσεων (Σχήμα 15.5) είναι εμφανές πως υπάρχει και δεύτερη λύση, που δεν εντοπίστηκε με την εντολή *solve*. Επομένως, θα πρέπει να είμαστε προσεκτικοί όταν επιλύουμε μη γραμμικές αλγεβρικές εξισώσεις και συστήματα με τον τρόπο αυτό, αφού υπάρχει ο κίνδυνος να μην εντοπίσουμε όλες τις λύσεις.



ΣΧΗΜΑ 15.5: ΓΡΑΦΙΚΗ ΠΑΡΑΣΤΑΣΗ ΤΩΝ ΔΥΟ ΜΗ ΓΡΑΜΜΙΚΩΝ ΑΛΓΕΒΡΙΚΩΝ ΕΞΙΣΩΣΕΩΝ.

Πέραν των προαναφερθέντων, η εντολή *solve* μπορεί να χρησιμοποιηθεί ακόμη πιο άμεσα, χωρίς να απαιτείται ο ορισμός κάποιας συμβολικής μεταβλητής. Στο παράδειγμα που ακολουθεί επιδεικνύεται η χρήση της εντολής *solve*.

```
[x,y] = solve('x^2 + x*y + y = 3','x^2 - 4*x + 3 = 0')
 x =
 [ 1]              % ορίζουμε απ' ευθείας τις εξισώσεις
 [ 3]              % μέσα σε απλά εισαγωγικά
 y =
 [    1]
 [ -3/2]
```

15.4.5 ΥΠΟΛΟΓΙΣΜΟΣ ΠΑΡΑΓΩΓΩΝ ΣΥΜΒΟΛΙΚΩΝ ΣΥΝΑΡΤΗΣΕΩΝ

Ορίζοντας μια συμβολική συνάρτηση, μπορούμε να υπολογίσουμε την νιοστή παράγωγό της με τη χρήση της εντολής *diff()*. Στα παραδείγματα που ακολουθούν υπολογίζεται η $1^{η}$ και $3^{η}$ παράγωγος μιας συνάρτησης.

```
>> syms x
>> fx=x^3 -2*x^2 -3*x + 10;

>> dfdx=diff(fx,1)
dfdx =
3*x^2-4*x-3

>> d3fdx3=diff(fx,3)
d3fdx3 =
6
```

Με τη χρήση της ίδιας εντολής δύναται να υπολογιστεί η μερική παράγωγος, όταν η συνάρτηση έχει δύο και πλέον μεταβλητές, ως ακολούθως:

```
>> syms x y
>> fxy=x^3 + y*x^2 + 15*x*y - 12;  % ορισμός συνάρτησης
>> diff(fxy,x)      % πρώτη παράγωγος ως προς x
ans =               % το y αντιμετωπίζεται ως σταθερά
3*x^2+2*x*y+15*y
>> diff(fxy,x,2)    % δεύτερη παράγωγος ως προς x
ans =
6*x+2*y
>> diff(fxy,x,y)  % πρώτη παράγωγος ως προς x και y
ans =
x^2+15*x
```

15.4.6 ΥΠΟΛΟΓΙΣΜΟΣ ΟΛΟΚΛΗΡΩΜΑΤΩΝ ΣΥΜΒΟΛΙΚΩΝ ΣΥΝΑΡΤΗΣΕΩΝ

Ο υπολογισμός ολοκληρωμάτων συμβολικών συναρτήσεων μπορεί να γίνει με τη χρήση της εντολής *int()*. Με την εντολή αυτή μπορούμε να υπολογίσουμε και ορισμένα, άλλα και αόριστα ολοκληρώματα.

```
>> syms x
>> fx=x^2+5*x+5;   % ορισμός συνάρτησης

>> int(fx)         % υπολογισμός αόριστου ολοκληρώματος
ans =
1/3*x^3+5/2*x^2+5*x

 >> int(fx,x)           % υπολογισμός αόριστου
                        % ολοκληρώματος ως προς τη
 ans =                  % μεταβλητή x (σε περίπτωση που
                        % έχουμε συνάρτηση
 1/3*x^3+5/2*x^2+5*x     % με περισσότερες των δύο
                         % μεταβλητών)

 >> int(fx,-1,5)         % υπολογισμός ορισμένου
                         % ολοκληρώματος από -1 ως 5
 ans =
 132
```

15.4.7 ΕΠΙΛΥΣΗ ΓΡΑΜΜΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΜΕ ΠΑΡΑΜΕΤΡΟΥΣ

Σε προηγούμενη ενότητα παρουσιάστηκε ο τρόπος βάσει του οποίου δύναται να επιλυθεί ένα σύστημα γραμμικών εξισώσεων. Ωστόσο, σε πολλές περιπτώσεις γραμμικών συστημάτων, δεν είναι γνωστός κάποιος ή κάποιοι από τους συντελεστές των σταθερών όρων, με αποτέλεσμα το σύστημα να καθίσταται παραμετρικό. Στην περίπτωση αυτή μπορούμε να ορίσουμε την άγνωστη αυτή παράμετρο (ή τις άγνωστες παραμέτρους) ως συμβολική μεταβλητή και να προχωρήσουμε σε επίλυση. Στο παρακάτω παράδειγμα παρουσιάζεται αυτή η μέθοδος επίλυσης.

```
>> syms k

>> A=[3,k,-1;-1,3,2;1,-1,-1]  % σύστημα με A(1,2)=k

A =
[  3,  k, -1]
[ -1,  3,  2]
[  1, -1, -1]

>> b=[10;5;-1];

>> det(A)              % υπολογίζουμε την ορίζουσα
ans =                  % αν το k≠1 το σύστημα έχει
-1+k                   % μοναδική λύση

>> x=inv(A)*b          % επίλυση του συστήματος, με χρήση
                       % αντιστρόφου πίνακα
x =
  -10/(-1+k)+5*(k+1)/(-1+k)-(2*k+3)/(-1+k)
                                  5 /(-1+k)
    -20/(-1+k)+5*(k+3)/(-1+k)-(k+9)/(-1+k)

>> simplify(x)         % το παραπάνω αποτέλεσμα μπορεί να
                       % απλοποιηθεί
 ans =
    (-8+3*k)/(-1+k)
          5/(-1+k)
 2*(-7+2*k)/(-1+k)

>> x=A\b              % απευθείας επίλυση με διαίρεση
                       % πινάκων
 x =
   (-8+3*k)/(-1+k)
          5/(-1+k)
 2*(-7+2*k)/(-1+k)

>> subs(x,k,2)        % υπολογίζουμε τη λύση για μία τιμή,
                       % π.χ. k=2,
ans =
    -2
     5
    -6
```

15.5 ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΣ FOURIER

Ο μετασχηματισμός Fourier αναλύει μία συνάρτηση σε άθροισμα τριγωνομετρικών συναρτήσεων. Μία υποκατηγορία του μετασχηματισμού Fourier είναι ο διακριτός μετασχηματισμός Fourier ο οποίος πραγματοποιείται με χρήση του αλγορίθμου Fast Fourier Transformation. Για μία δεδομένη χρονοσειρά (μετρήσεις ανά τακτά χρονικά διαστήματα) η οποία αποτελείται από N στοιχεία, ο FFT μας δίνει ένα N – διάστατο διάνυσμα με βάση την σχέση

$$x(n) = \frac{1}{N}\sum_{k=1}^{N}\left[a(k)\cos\left(\frac{2\pi(k-1)(n-1\ )}{N}\right) + b(k)\sin\left(\frac{2\pi(k-1)(n-1\ )}{N}\right)\right]$$

Η σχέση που προκύπτει από τον FFT είναι λοιπόν μία μιγαδική συνάρτηση[33] , που χάριν ευκολίας την ονομάζουμε συνάρτηση Y. Με τη βοήθεια λοιπόν του μετασχηματισμού αυτού, μία χρονοσειρά αναλύεται σε αθροίσματα όρων τριγωνομετρικού χαρακτήρα, δηλ. όρων με περιοδικότητα, γεγονός που μπορεί να μας επιτρέψει να καταδείξουμε τις τυχόν περιοδικότητες που υπάρχουν εντός της ίδιας της χρονοσειράς. Στο παράδειγμα που ακολουθεί αποσαφηνίζεται περισσότερο η λειτουργία του FFT, καθώς και ο τρόπος με τον οποίο μπορούμε να τον εφαρμόσουμε, χρησιμοποιώντας τη συνάρτηση *fft()* του MATLAB.

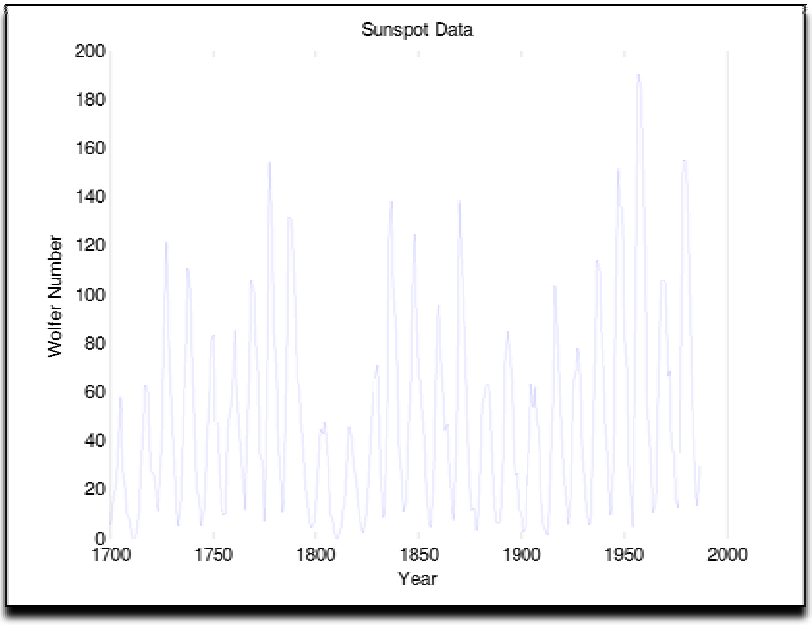
- **Παράδειγμα: περιοδικότητα του αριθμού των ηλιακών κηλίδων**

Στο παρόν παράδειγμα εφαρμογής γίνεται χρήση του αρχείου sunspot.dat, το οποίο βρίσκεται στο MATLAB, και συνίσταται από δύο στήλες: Στη πρώτη στήλη περιλαμβάνονται τα έτη από 1700 μέχρι και το 1987, ενώ στη δεύτερη ο αντίστοιχος αριθμός Wolfer. Ο αριθμός αυτός είναι ενδεικτικός για την ηλιακή δραστηριότητα (τον αριθμό των κηλίδων στην επιφάνεια του ήλιου). Το αρχείο αρχικά «φορτώνεται» στο MATLAB με τη βοήθεια της εντολής *load*. Αμέσως μετά δημιουργείται η γραφική παράσταση του αριθμού Wolfer συναρτήσει του χρόνου, με χρήση των ακόλουθων εντολών

```
>> load sunspot.dat
year = sunspot(:,1);
```

[33] Υπενθυμίζεται εδώ η σχέση του De'Moivre που συνδέει τριγωνομετρικές με μιγαδικές συναρτήσεις: $(cosx + isinx)^n = \cos(nx) + isin(nx)$.

```
wolfer = sunspot(:,2);
plot(year,wolfer)
xlabel('Year')
ylabel('Wolfer Number')
title('Sunspot Data')
```



ΣΧΗΜΑ 15.6: ΓΡΑΦΙΚΗ ΠΑΡΑΣΤΑΣΗ ΤΟΥ ΑΡΙΘΜΟΥ WOLFER ΣΥΝΑΡΤΗΣΕΙ ΤΟΥ ΧΡΟΝΟΥ.

Από το Σχήμα 15.6 είναι εμφανές ότι υπάρχει κάποια περιοδικότητα στην ηλιακή δραστηριότητα, χωρίς ωστόσο να είναι εύκολο να προσδιορίσουμε την συχνότητα επανάληψης του φαινομένου. Αυτό οφείλεται στο γεγονός ότι υπάρχουν περισσότερες από μία συχνότητες, καθώς επίσης και στο ότι οι μετρήσεις δεν είναι απόλυτα «ακριβείς», υπάρχει δηλαδή «θόρυβος» στα δεδομένα. Γενικότερα, για ένα διακριτό σύνολο μετρήσεων συναρτήσει του χρόνου (διακριτή χρονοσειρά), το ανώτατο όριο της συχνότητας, (δηλαδή η μικρότερη περίοδος, μια και ————) για το οποίο η έννοια της περιοδικότητας έχει νόημα, είναι ένας κύκλος για κάθε δύο επιτυχείς μετρήσεις, και ονομάζεται συχνότητα Nyquist[34].

[34] Η χρήση στο παράδειγμά μας της μεταβλητής **`nyquist`** δεν πρέπει να συγχέεται με την ομώνυμη εντολή του MATLAB.

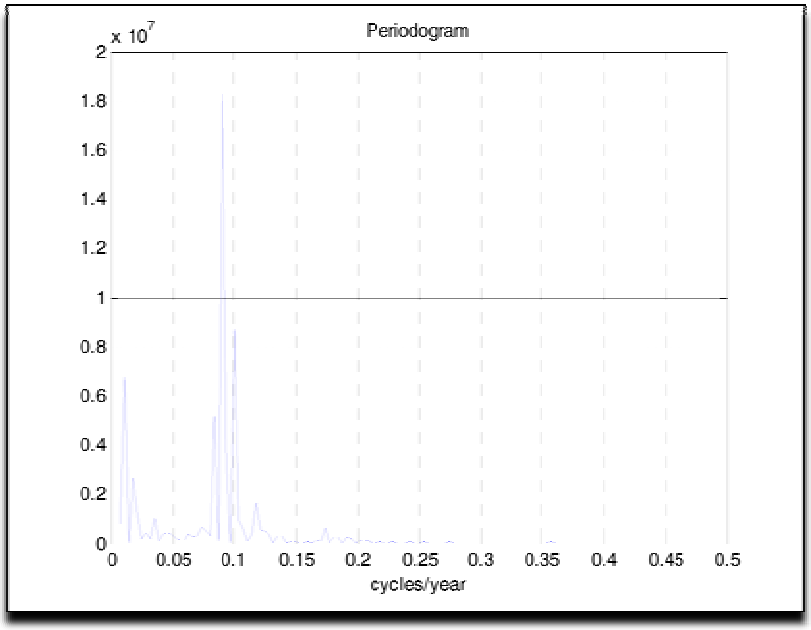
Εδώ λοιπόν, και σε απάντηση του ερωτήματος σε σχέση με το ποιες συχνότητες είναι σημαντικές για την υπό μελέτη χρονοσειρά, εισάγεται η χρήση των περιοδογραμμάτων. Τα περιοδογράμματα είναι μια μέθοδος βάσει της οποίας μπορούμε να ανακαλύψουμε τις «κρυμμένες» αρμονικές συναρτήσεις (άρα και περιόδους) μίας συνάρτησης. Το περιοδόγραμμα εξετάζει όλες τις δυνατές συχνότητες και ποσοτικοποιεί την σπουδαιότητα της καθεμίας για τη χρονοσειρά που μελετούμε. Όπως έχουμε ήδη δει, εφαρμόζοντας τον FFT σε μία χρονοσειρά, λαμβάνουμε ως αποτέλεσμα τη συνάρτηση Y. Το τετράγωνο του μέτρου της Y ονομάζεται ισχύς (συμβολίζεται και ως r^2), και το διάγραμμα της ισχύος ως προς την συχνότητα αποτελεί το περιοδόγραμμα. Τα σημεία εμφάνισης μεγίστων στο σχετικό διάγραμμα αποτελούν ένδειξη περιοδικότητας.

Όλα αυτά θα γίνουν άμεσα αντιληπτά με την εφαρμογή του FFT στα δεδομένα του παραδείγματος. Οι απαραίτητες εντολές παρουσιάζονται και επεξηγούνται παρακάτω, ενώ το περιοδόγραμμα που προκύπτει βρίσκεται στο Σχήμα 15.7.

```
Y = fft(wolfer);
N = length(Y);

% η length δίνει το μήκος (σε αριθμό στοιχείων)
% του διανύσματος N

Y(1) = [];
% η εντολή αυτή αφαιρεί από το διάνυσμα N το πρώτο στοιχείο
% Αυτό γίνεται διότι ο FFT παράγει μία σειρά μιγαδικών, και
% έναν πραγματικό, τον οποίο τοποθετεί πρώτο.
% Ο αριθμός αυτός πρέπει να απαλειφθεί, διότι ως σταθερή
% ποσότητα, δεν έχει περιοδικότητα.

power = abs(Y(1:N/2)).^2;

% υπολογισμός διανύσματος ισχύος για τα μισά (N/2)
% από τα στοιχεία του Y
% Προσέξτε ότι κάθε στοιχείο του Y που χρησιμοποιείται
% υψώνεται στο τετράγωνο.

nyquist = 1/2;

freq = (1:N/2)/(N/2)*nyquist;
```

```
% δημιουργία του διανύσματος freq με τόσα στοιχεία όσα
% και τα στοιχεία του power, και τιμές από 0 έως τη
% συχνότητα nyquist, δηλ. ½

plot(freq,power), grid on
xlabel('cycles/year')
title('Periodogram')
```

Χωρίς να μπούμε σε περισσότερες λεπτομέρειες για τις εντολές που ακολουθούν την βασική εντολή εφαρμογής του FFT (*Y=fft(wolfer)*), παρατηρούμε από το Σχήμα 15.7 πως η βασική συχνότητα από την οποία συνίσταται η περιοδικότητα της χρονοσειράς είναι η . Μετατρέποντας την συχνότητα στην αντίστοιχη περίοδο, διαπιστώνουμε ότι η περίοδος επανάληψης του φαινομένου είναι , δηλαδή κάθε περίπου 11 έτη έχουμε έξαρση στην ηλιακή δραστηριότητα.



ΣΧΗΜΑ 15.7: ΠΕΡΙΟΔΟΓΡΑΜΜΑ ΤΗΣ ΗΛΙΑΚΗΣ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ.

15.6 ΔΙΑΦΟΡΙΚΕΣ ΕΞΙΣΩΣΕΙΣ

Οι διαφορικές εξισώσεις (Δ.Ε.) αποτελούν μια μαθηματική έννοια που βρίσκει εφαρμογή σε όλους τους κλάδους της σύγχρονης επιστήμης. Η κεντρική ιδέα μιας διαφορικής εξίσωσης είναι πολύ απλή. Αναφέρεται σε κάθε σχέση που συνδέει μια μεταβλητή x, μια άγνωστη συνάρτηση $y(x)$ και τις παραγώγους της $y'(x), y''(x)$ κτλ. Επομένως, μια διαφορική εξίσωση μπορεί να γραφεί ως εξής:

$$F(x, y(x), y'(x), y''(x), \dots) = 0$$

Η επίλυση μιας διαφορικής εξίσωσης συνίσταται στην εύρεση της άγνωστης συνάρτησης $y(x)$. Η διαδικασία αυτή δεν είναι συνήθως εύκολη, ενώ στις πιο περίπλοκες περιπτώσεις είναι αδύνατη, οπότε και καταφεύγουμε σε αριθμητικές μεθόδους επίλυσης.

Μία διαφορική εξίσωση είναι σε θέση να περιγράψει τη δυναμική – χρονική εξέλιξη ενός συστήματος και άρα μπορεί να χρησιμοποιηθεί για να προβλέψουμε την εξέλιξη και τη συμπεριφορά του. Το επιπλέον στοιχείο που απαιτείται να γνωρίζουμε, πέρα από τη διαφορική εξίσωση αυτή καθεαυτή, είναι κάποια αρχική συνθήκη, δηλαδή να γνωρίζουμε με ακρίβεια την κατάσταση του συστήματος μία συγκεκριμένη χρονική στιγμή, που συνήθως είναι η στιγμή που «εκκινεί» το δυναμικό φαινόμενο. Έχοντας τη γνώση αυτή το σύστημα είναι απόλυτα προσδιορισμένο. Επιρπόσθετα, πολλές φορές είναι χρήσιμο να γνωρίζουμε τις οριακές συνθήκες, δηλ. τις τιμές που λαμβάνουν οι μεταβλητές του προβλήματος σε σχέση με όριά τους (φυσικά, χρονικά κλπ).

Από τα παραπάνω γίνεται εμφανές το πόσο σημαντικό είναι να γνωρίζουμε την διαφορική εξίσωση ενός συστήματος, αλλά και να μπορούμε να την επιλύουμε. Στο MATLAB μας δίνεται η δυνατότητα να επιλύουμε μια διαφορική εξίσωση με διάφορες μεθόδους. Η πιο απλή από αυτές έχει να κάνει με τη συμβολική επίλυσή της, και επιτυγχάνεται με τη χρήση της εντολής *dsolve()*.

- **Παραδείγμα Ι: επίλυσης της πρωτοτάξιας Δ.Ε. $y'(x) = xy$**

Έστω $y(x)$μία συνάρτηση μιας μεταβλητής, και έστω η πρωτοτάξια διαφορική εξίσωση

$$y'(x) = xy$$

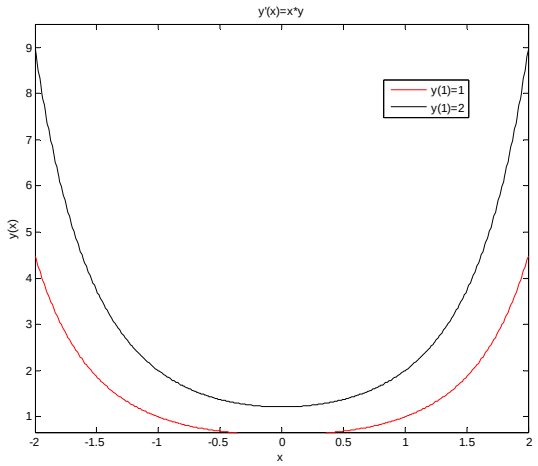
Ο ορισμός και η συμβολική επίλυσή της στο MATLAB γίνεται ως ακολούθως:

```
>> diffequation='Dy=x*y'  % ορισμός της διαφορικής εξίσωσης
diffequation =            % με D συμβολίζεται η παράγωγος
Dy=x*y

>> y=dsolve(diffequation,'x')
% επίλυση χωρίς αρχικές συνθήκες
y =
C1*exp(1/2*x^2)
```

Στην λύση της παραπάνω εξίσωσης βλέπουμε ότι εμφανίζεται μία «αυθαίρετη» σταθερά, η C_1. Ο προσδιορισμός αυτής μπορεί να γίνει με χρήση οριακών συνθηκών. Έτσι, εάν για παράδειγμα είναι $y(1) = 1$, τότε μπορούμε να λύσουμε την εξίσωση ως εξής:

```
>> initialcondition='y(1)=1'    % ορισμός αρχικής συνθήκης
initialcondition =
y(1)=1

>> y=dsolve(diffequation,initialcondition,'x')
y =                             % επίλυση με αρχική συνθήκη
1/exp(1/2)*exp(1/2*x^2)
```

Η λύση στην περίπτωση αυτή παρατηρούμε ότι είναι πλήρως προσδιορισμένη και μπορούμε να την απεικονίσουμε γραφικά. Η μορφή στην οποία έχουμε τη λύση είναι συμβολική, άρα για την γραφική της παράσταση χρησιμοποιούμε την εντολή *ezplot()*, η οποία κατασκευάζει τη γραφική παράσταση της συνάρτησης που θα τοποθετηθεί εντός της παρένθεσης, για το διάστημα -2π < x < 2π. Στο Σχήμα 15.8 παρουσιάζεται η γραφική απεικόνιση της λύσης, για δύο διαφορετικές αρχικές συνθήκες.



ΣΧΗΜΑ 15.8: ΓΡΑΦΙΚΗ ΑΠΕΙΚΟΝΙΣΗ ΤΗΣ ΛΥΣΗΣ ΤΗΣ ΔΙΑΦΟΡΙΚΗΣ ΕΞΙΣΩΣΗΣ $y'(x) = xy$. Η ΑΝΩ ΚΑΜΠΥΛΗ ΑΝΤΙΣΤΟΙΧΕΙ ΣΕ Υ(1)=1 ΚΑΙ Η ΚΑΤΩ ΣΕ Υ(1)=2

- **Παράδειγμα ΙΙ: Επίλυση της δευτεροτάξιας δ. ε.**
  $\boldsymbol{y'' + 8y' + 2y = \cos(x)}$

Οι διαφορικές εξισώσεις που είδαμε μέχρι τώρα χαρακτηρίζονται ως πρωτοτάξιες διαφορικές εξισώσεις, γιατί σε αυτές εμφανίζονται μόνο οι πρώτες παράγωγοι των συναρτήσεων. Σε περίπτωση που έχουμε να επιλύσουμε διαφορικές εξισώσεις δεύτερης ή μεγαλύτερης τάξης, ακολουθούμε τον ίδια ακριβώς μέθοδο. Έτσι, αν έχουμε τη δευτεροτάξια διαφορική εξίσωση

$$y'' + 8y' + 2y = \cos(x) \qquad (15.7)$$

τότε η λύση θα προκύψει από εντολές όπως οι ακόλουθες:

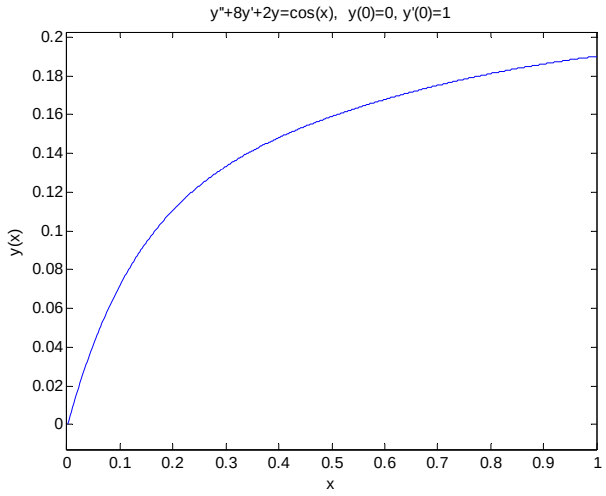
```
>> diffequation='D2y+8*Dy+2*y=cos(x)'
% με D2 συμβολίζεται η δεύτερη παράγωγος

diffequation =
D2y+8*Dy+2*y=cos(x)
>> y=dsolve(diffequation,'x')
y =
exp(-4*x+x*14^(1/2))*C2+exp(-4*x-
x*14^(1/2))*C1+1/65*cos(x)+8/65*sin(x)
```

Παρατηρούμε στην περίπτωση αυτή ότι η λύση είναι αρκετά πολύπλοκη, καθώς και ότι εμφανίζονται σε αυτή δύο σταθερές, οι C_1 και C_2 (ο αριθμός των σταθερών ισούται με την τάξη της εξίσωσης). Έτσι, για τον προσδιορισμό τους χρειαζόμαστε δύο αρχικές συνθήκες. Ας υποθέσουμε πως αυτές είναι οι: $y(0) = 0,\ y'(0) = 1$. Βάσει αυτών, η λύση σε MATLAB προκύπτει με χρήση μιας ομάδας εντολών όπως η παρακάτω:

```
>> init='y(0)=0, Dy(0)=1'
arxikes =
y(0)=0, Dy(0)=1

>> y=dsolve(diffequation,arxikes,'x')
 y =
 exp((-4+14^(1/2))*x)*(53/1820*14^(1/2)-1/130)+exp(-
(4+14^(1/2))*x)*(-1/130-
53/1820*14^(1/2))+1/65*cos(x)+8/65*sin(x)
```

Η λύση αυτή μπορεί να οπτικοποιηθεί κάνοντας χρήση της εντολής *ezplot()*, όπως παρουσιάζεται στο Σχήμα 15.9.

Τέλος, θα πρέπει να σημειώσουμε ότι η εντολή *dsolve()* δεν είναι σε θέση να μας δώσει τη λύση όταν η διαφορική εξίσωση είναι αρκετά πολύπλοκη. Στην περίπτωση αυτή μπορούμε να χρησιμοποιήσουμε μια αριθμητική μέθοδο επίλυσης, που υποστηρίζονται από το MATLAB .



ΣΧΗΜΑ 15.9: ΓΡΑΦΙΚΗ ΑΠΕΙΚΟΝΙΣΗ ΤΗΣ ΛΥΣΗΣ ΤΗΣ ΔΙΑΦΟΡΙΚΗΣ ΕΞΙΣΩΣΗΣ 15.7.