ΚΕΦΑΛΑΙΟ 5: ΠΡΟΣΕΓΓΙΣΗ ΣΥΝΑΡΤΗΣΕΩΝ ΚΑΙ ΔΕΔΟΜΕΝΩΝ ΜΕ ΤΗΝ ΜΕΘΟΔΟ ΤΩΝ ΕΛΑΧΙΣΤΩΝ ΤΕΤΡΑΓΩΝΩΝ

Περιεχόμενα

1. Πολυωνυμική προσέγγιση

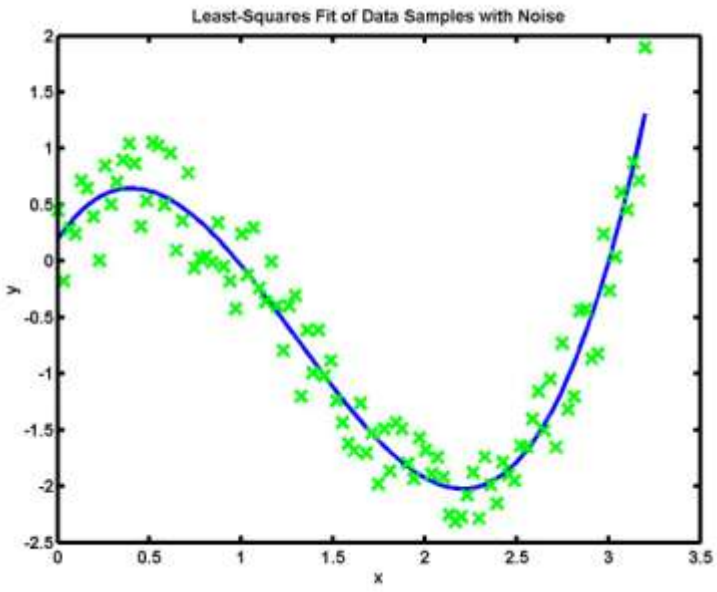
Πρόβλημα: Δεδομένου ενός συνόλου ***(n+1)*** σημείων:

$$(x_1,y_1),(x_2,y_2), \ldots, (x_n,y_n), (x_{n+1},y_{n+1})$$

Βρείτε ένα πολυώνυμο βαθμού $m < n$, $y = \Phi_m(x)$, που «ταιριάζει – προσεγγίζει – προσαρμόζει» (fit) καλύτερα το σύνολο δεδομένων σημείων ***(n+1).***

Παρατήρηση: Εάν το σύνολο δεδομένων έχει επηρεαστεί από μη ακριβείς μετρήσεις, εξαιτίας θορύβου ή αν οι τιμές των δεδομένων στρογγυλοποιηθούν, η πολυωνυμική παρεμβολή παράγει μια πολύ δύσκαμπτη καμπύλη με απότομες γωνίες και μεγάλο λάθος παρεμβολής. Στην περίπτωση αυτή, αντί της πολυωνυμικής παρεμβολής χρησιμοποιείται μια άλλη τεχνική που βασίζεται στην **αριθμητική προσέγγιση** των τιμών των δεδομένων σημείων.

Η γραφική παράσταση που ακολουθεί μας δείχνει την αριθμητική προσέγγιση περισσοτέρων από εκατό δεδομένα από ένα πολυώνυμο 3ου βαθμού:



Συνήθως, η πολυωνυμική προσέγγιση είναι ικανοποιητική όταν οι τιμές των δεδομένων προσαρμόζονται από ένα πολυώνυμο μικρού βαθμού. Σε πολλές περιπτώσεις, οι απλές (πολυωνυμικές) καμπύλες προβλέπονται από λύσεις θεωρητικών μοντέλων (π.χ. διαφορικές εξισώσεις). Οι μέθοδοι αριθμητικών προσεγγίσεων επιτρέπουν στους ερευνητές να συγκρίνουν θεωρητικά μοντέλα με δείγματα πραγματικών δεδομένων.

Μέθοδοι:

- Ελαχιστοποίηση του τετραγώνου του ολικού σφάλματος
- Υπερ-προσδιορισμένο γραμμικό σύστημα
- MATLAB **"polyfit"** συναρτήσεις

2. Ελαχιστοποίηση του τετραγώνου του ολικού σφάλματος

Ας υποθέσουμε ότι $y = \Phi_m(x)$ είναι ένα πολυώνυμο m βαθμού, το οποίο προσαρμόζεται στα δεδομένα σημεία ελαχιστοποιώντας το τετράγωνο του ολικού σφάλματος μεταξύ των δεδομένων σημείων και των τιμών του:

$$\boxed{(min)\ E = \sum_{j=1}^{n+1} \left(y_j - \Phi_m(x_j) \right)^2}$$

Στην περίπτωση της γραμμικής παλινδρόμησης, δηλαδή όταν η συνάρτηση Φ είναι πολυώνυμο βαθμού 1, $y = \Phi_1(x) = c_1 x + c_2$

το ολικό σφάλμα είναι: $E = \sum_{j=1}^{n+1} \left(y_j - c_1 x_j - c_2 \right)^2$

Οι συνθήκες ελαχιστοποίησης του ολικού σφάλματος δίδονται από το σύστημα:

$$\frac{\partial E}{\partial c_1} = -2 \sum_{j=1}^{n+1} x_j \left(y_j - c_1 x_j - c_2 \right) = 0$$

$$\frac{\partial E}{\partial c_2} = -2 \sum_{j=1}^{n+1} \left(y_j - c_1 x_j - c_2 \right) = 0$$

του οποίου η λύση προσδιορίζει τους συντελεστές του γραμμικού πολυωνύμου.

Η μέθοδος ελαχίστων τετραγώνων (ΕΤ) χρησιμοποιείται για να προσδιορίζει τις παραμέτρους μιας δεδομένης οικογένειας καμπυλών έτσι ώστε να ταιριάζουν καλύτερα σε ένα σύνολο δεδομένων. Η χρήση της δικαιολογείται για δυο τουλάχιστον λόγους:

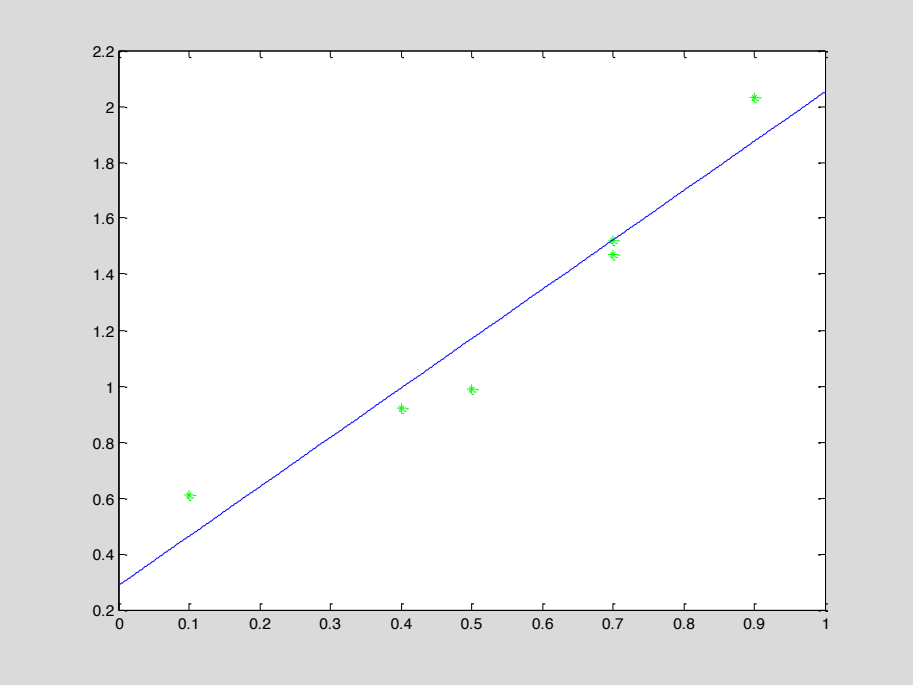
a) Υπάρχουν φορές που έχουμε ένα σύνολο δεδομένων το οποίο είναι κατάλληλο για παρεμβολή, αλλά έχουμε αποφασίσει ότι δε χρειαζόμαστε τόση ακρίβεια όση μας προσφέρει η παρεμβολή. Δεδομένων των επιτρεπτών ορίων αβεβαιότητας στους υπολογισμούς μας, μπορεί να είμαστε ικανοποιημένοι με μια απλή συνάρτηση (πιθανώς πολυωνυμική) που περνάει κοντά από τα σημεία των δεδομένων μας αλλά όχι ακριβώς πάνω από όλα αυτά.
b) Σε πολλές πειραματικές διαδικασίες, τα σημεία δεδομένων μας περιέχουν ένα σφάλμα μέτρησης και δεν μπορούν να χρησιμοποιηθούν άμεσα για να γίνουν προβλέψεις. Στην περίπτωση αυτή, βάσει κάποιων θεωρήσεων, προσδιορίζουμε τη γενική μορφή μιας συνάρτησης που θα έπρεπε να ταιριάζει στη συμπεριφορά των δεδομένων. Μετά, εφαρμόζουμε τη μέθοδο των ελαχίστων τετραγώνων για να επιλέξουμε τις τιμές των απροσδιόριστων παραμέτρων της επιλεγμένης συνάρτησης.

Το παρακάτω MatLab πρόγραμμα υλοποιεί την παραπάνω μέθοδο για ένα σύνολο σημείων.

```
x = [ 0.1,0.4,0.5,0.7,0.7,0.9 ]
y = [ 0.61,0.92,0.99,1.52,1.47,2.03]
a11 = sum(x.^2); a12 = sum(x); a21 = sum(x); a22 =
sum(ones(1,length(x)));
A = [ a11,a12; a21,a22]; % the coefficient matrix of the minimization
problem
b1 = sum(x.*y); b2 = sum(y);
b = [ b1; b2 ]; % right-hand-side of the minimization problem
c = A \ b  % solution of the minimization problem
xApr = 0 : 0.001 : 1; yApr = c(1)*xApr + c(2);
plot(x,y,'*g',xApr,yApr,'b');
```

```
x =    0.1000    0.4000    0.5000    0.7000    0.7000    0.9000
y =    0.6100    0.9200    0.9900    1.5200    1.4700    2.0300
```

```
c =    1.7646    0.2862
```



3. Υπέρ-προσδιοριμένα γραμμικά συστήματα

Ας υποθέσουμε ότι το πολυώνυμο προσέγγισης $y = \Phi_m(x)$ περνάει απ' όλα τα δεδομένα σημεία: $\Phi_m(x_j) = y_j$. Στην περίπτωση αυτή ($m<n$), οι συνθήκες παρεμβολής ορίζουν ένα υπερ-προσδιορισμένο γραμμικό σύστημα $A\ c = b$ (ο αριθμός των εξισώσεων $(n+1)$ υπερβαίνει τον αριθμό των αγνώστων $(m+1)$). Εφόσον, ο πίνακας των συντελεστών A είναι μη – αντιστρέψιμος (singular), δεν υπάρχει λύση ενός τέτοιου υπερ-προσδιορισμένου συστήματος. Ωστόσο, μπορούμε να λύσουμε το σύστημα αν το μετασχηματίσουμε πολλαπλασιάζοντας με τον ανάστροφο του A τα δύο μέρη του συστήματος: $(A'*A)\ c = (A'*b)$. Ο πίνακας συντελεστών $(A'*A)$ είναι πλέον ένας 2 επι 2 αντιστρέψιμος πίνακας έτσι ώστε να υπάρχει μια μοναδική λύση για το c. Αυτή η λύση είναι η ίδια με αυτή που δίνει το ελάχιστο του τετραγώνου του ολικού σφάλματος E.

Γραμμική παλινδρόμηση (Linear regression)

$$c_1 x_j + c_2 = y_j, \quad j = 1,2,\ldots,(n+1)$$

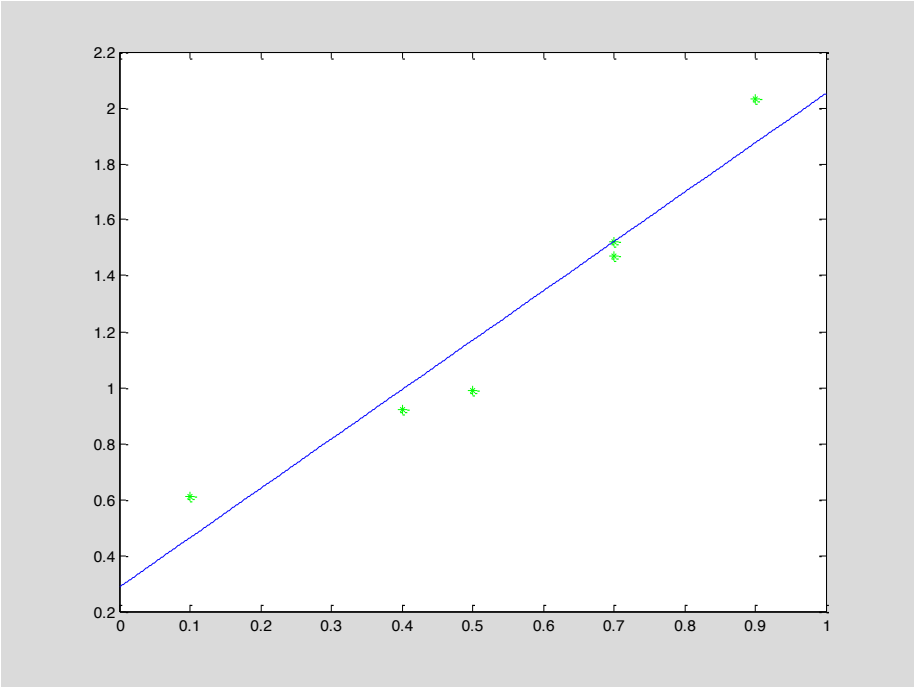
ως λύση του συστήματος $(A'*A)\ c = (A'*b)$.

Η υλοποίηση της παραπάνω λύσης δίδεται από τον παρακάτω MatLab κώδικα:

```
x = [ 0.1,0.4,0.5,0.7,0.7,0.9 ];
y = [ 0.61,0.92,0.99,1.52,1.47,2.03];
A = [ x', ones(length(x),1) ];
   % the coefficient matrix of the over-determined problem
b = y'; % right-hand-side of the over-determined problem
c = (A'*A) \ (A'*b)  % solution of the over-determined problem
```

```
xApr = 0 : 0.001 : 1; yApr = c(1)*xApr + c(2);
plot(x,y,'*g',xApr,yApr,'b');
```

```
c =    1.7646    0.2862
```



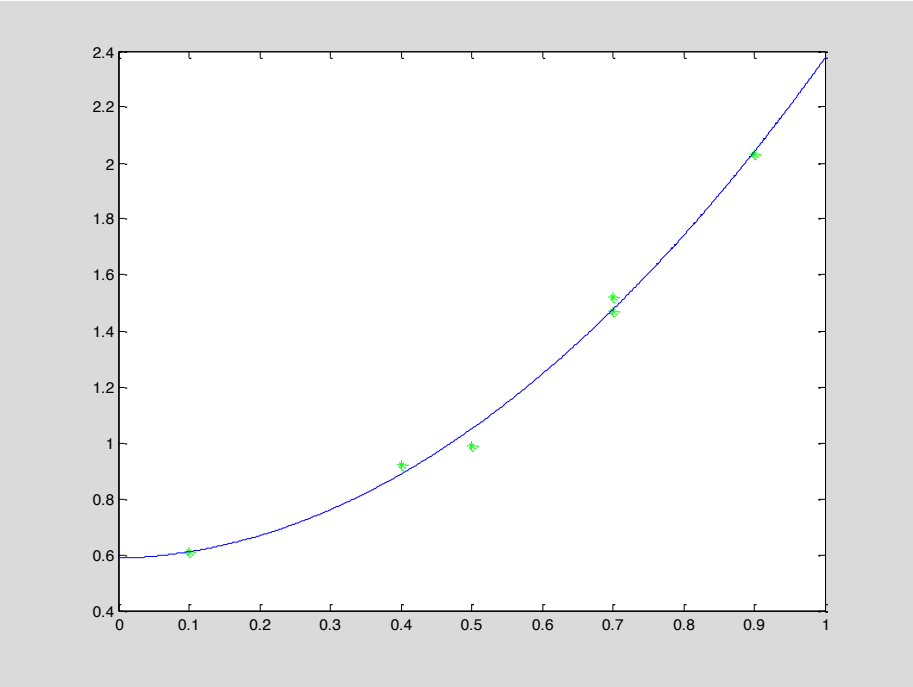

```
A, b, c = A\b
   % MATLAB solves the over-determined system in the least square sense
E = sum((y-c(1)*x-c(2)).^2)
```

```
A = 0.1000    1.0000     b = 0.6100              c = 1.7646
    0.4000    1.0000         0.9200                  0.2862
    0.5000    1.0000         0.9900
    0.7000    1.0000         1.5200              E = 0.0856
    0.7000    1.0000         1.4700
    0.9000    1.0000         2.0300
```

Το τετράγωνο του ολικού σφάλματος ***E*** γίνεται μικρότερο εάν ο βαθμός ***m < n*** του πολυώνυμο προσέγγισης ***y =*** $\Phi_m(x)$ αυξάνει**.** Το τετράγωνο του ολικού σφάλματος ***E*** είναι μηδέν, εάν ***m=n,*** δηλαδή το πολυώνυμο προσέγγισης ***y =*** $\Phi_m(x)$ είναι το πολυώνυμο παρεμβολής ***y =*** $P_n(x)$ που διέρχεται από όλα τα ***(n+1)*** σημεία δεδομένων. Ο παρακάτω κώδικας επιβεβαιώνει αυτή την παρατήρηση.

```
x = [ 0.1,0.4,0.5,0.7,0.7,0.9 ];
y = [ 0.61,0.92,0.99,1.52,1.47,2.03];
n = length(x)-1;
for m = 1 : n
A = vander(x); A = A(:,n-m+1:n+1);
   b = y'; c = A\b; yy = polyval(c,x);
E = sum((y-yy).^2)  ;
   fprintf('m = %d, E = %6.5f\n',m,E);
end
m = 2; A = vander(x); A = A(:,n-m+1:n+1); b = y';
c = A\b; xApr = 0 : 0.001 : 1; yApr = polyval(c,xApr);
plot(x,y,'*g',xApr,yApr,'b');
```

```
m = 1, E = 0.08564
m = 2, E = 0.00665
m = 3, E = 0.00646
m = 4, E = 0.00125
Warning: Matrix is singular to working precision.
m = 5, E =    Inf
```



Το σύστημα των γραμμικών εξισώσεων για την προσέγγιση της ελαχιστοποίησης του τετραγώνου ***(A'*A) c = (A'*b)*** είναι κακής κατάστασης όταν το ***n*** είναι μεγάλο καθώς και όταν τα σημεία δεδομένων περιλαμβάνουν συνδυασμό πολύ μικρών και πολύ μεγάλων τιμών για το ***x.*** Σ' αυτές τις περιπτώσεις πολύ συχνά χρησιμοποιούμε προσεγγίσεις με ορθογώνια πολυώνυμα.

- Για ***n+1*** δεδομένα**,** η συνάρτηση MatLab **polyfit** υπολογίζει τους συντελεστές πολυωνύμου προσέγγισης βαθμού ***m*** **με** την μέθοδο των ελαχίστων τετραγώνων όταν ***m<n*** και συντελεστές πολυωνύμων παρεμβολής όταν το ***m=n***

```
c = polyfit(x,y,1)
c =    1.7646    0.2862
```

4. Μη-γραμμική προσέγγιση

Πολλές φορές οι θεωρητικές εξαρτήσεις δεδομένων μπορούν να αναπαρασταθούν με μη-γραμμικές συναρτήσεις παρά με πολυώνυμα. Για παράδειγμα, οι θεωρητικές καμπύλες μπορούν να προσεγγιστούν από δυναμοσυναρτήσεις και εκθετικές συναρτήσεις. Στις περιπτώσεις αυτές, οι άξονες ***(x,y)*** μπορεί να επαναπροσδιοριστούν έτσι ώστε να μπορεί να χρησιμοποιηθεί γραμμικό πολυώνυμο για την προσέγγιση των μη-γραμμικών συναρτήσεων με την μέθοδο των ελαχίστων τετραγώνων.

α) Προσέγγιση δεδομένων με δυναμοσυναρτήσεις: $y = \beta x^{\alpha}$

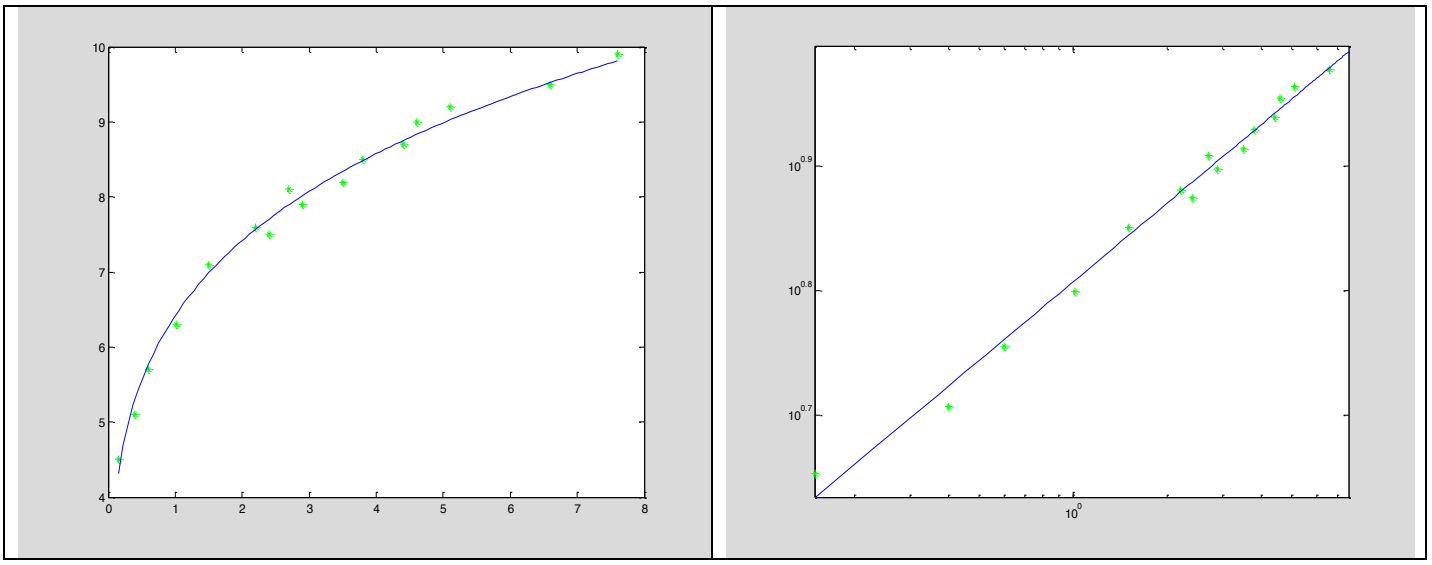
Στην περίπτωση αυτή, παίρνοντας το λογάριθμο της δυναμοσυνάρτησης μετασχηματίζουμε την συνάρτηση σε γραμμική στους άξονες ***(logx,logy)***:

$$\log y = \alpha \log x + \log \beta$$

και το δείγμα δεδομένων μπορεί να προσεγγιστεί με μια γραμμική παλινδρόμηση. Ο παρακάτω κώδικας δίνει ένα παράδειγμα της υλοποίησης της μεθόδου

```
x =
[0.15,0.4,0.6,1.01,1.5,2.2,2.4,2.7,2.9,3.5,3.8,4.4,4.6,5.1,6.6,7.6];
y = [4.5,5.1,5.7,6.3,7.1,7.6,7.5,8.1,7.9,8.2,8.5,8.7,9.0,9.2,9.5,9.9];
c = polyfit(log(x),log(y),1) % linear regression in logarithmic axis
xInt = linspace(x(1),x(length(x)),100);
yInt = exp(c(2))*xInt.^(c(1));
plot(x,y,'g*',xInt,yInt,'b');  % nonlinear regression in (x,y)
loglog(x,y,'g*',xInt,yInt,'b'); % linear regression in (logx,logy)

c =    0.2094    1.8588
```



β) Προσέγγιση δεδομένων με εκθετικές συναρτήσεις: $y = \beta e^{\alpha x}$

Στην περίπτωση αυτή παίρνουμε το λογάριθμο της εκθετικής συνάρτησης:

$$\log y = \alpha x + \log \beta$$

Σε ημιλογαριθμικούς άξονες ***(x,logy),*** το δείγμα δεδομένων μπορεί να προσεγγιστεί με γραμμικό πολυώνυμο.

γ) Προσέγγιση με γραμμικό συνδυασμό μη-γραμμικών συναρτήσεων

$$y = c_1 f_1(x) + c_2 f_2(x) + \ldots + c_n f_n(x)$$

Η γενική μη-γραμμική προσέγγιση $y = f(x;c_1,c_2,\ldots,c_n)$ αντιστοιχεί στην λύση ενός πολύπλοκου μη γραμμικού προβλήματος βελτιστοποίησης.

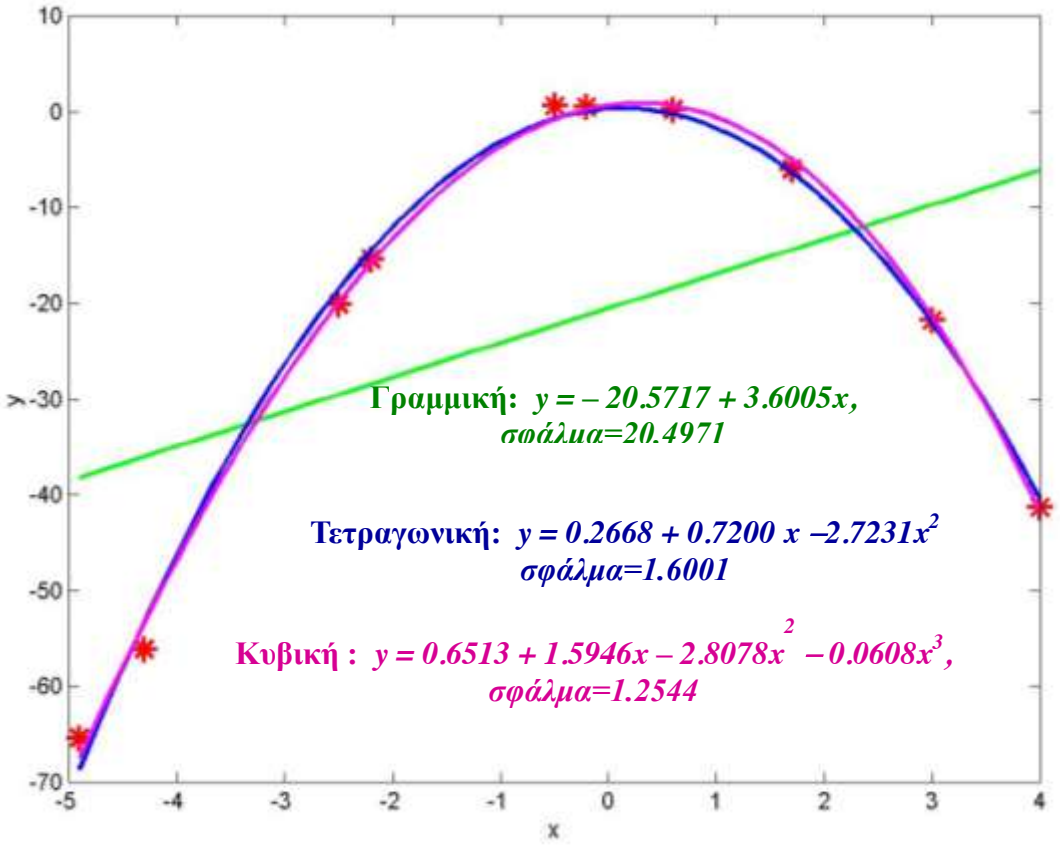
5. Παραδείγματα χρήσης της Matlab συνάρτησης polyfit

Παράδειγμα: Ο παρακάτω κώδικας προσαρμώζει τα πολυώνυμα 1ου, 2ου, και 3ου βαθμού στα παρακάτω δεδομένα, υπολογίζει το σφάλμα τους

$$E_2(f) = (\frac{1}{n}\sum_{k=1}^{m}|\mathrm{f}(\mathrm{x}_k) - \mathrm{y}_k|^2)^{1/2}$$

και παράγει την γραφική παράσταση δεδομένων και πολυωνύμων.

```
x = [ -2.5   3.0  1.7  -4.9 0.6 -0.5   4.0  -2.2  -4.3 -0.2];
y = [-20.1 -21.8 -6.0 -65.4 0.2  0.6 -41.3 -15.4 -56.1  0.5];
n=length(x);
sx=sum(x);sx2=sum(x.^2);
x1=min(x); x2=max(x); xx=x1:(x2-x1)/100:x2;
c1=polyfit(x,y,1);
ny1=polyval(c1,xx);
c2=polyfit(x,y,2);
ny2=polyval(c2,xx);
c3=polyfit(x,y,3);
ny3=polyval(c3,xx);
e1=sqrt( sum(abs(polyval(c1,x)-y).^2 )/n )
e2=sqrt( sum(abs(polyval(c2,x)-y).^2 )/n )
e3=sqrt( sum(abs(polyval(c3,x)-y).^2 )/n )
H=plot(x,y,'r*',xx,ny1,'g',xx,ny2,'b',xx,ny3,'m');
xlabel('x'); ylabel('y');
set(H,'LineWidth',3,'MarkerSize',12);
```



6. Παραδείγματα χρήσης της συνάρτησης polyfit Python

Παράδειγμα: Ο παρακάτω κώδικας προσαρμώζει τα πολυώνυμα 1ου, 3ου, και 30ου βαθμού στα παρακάτω δεδομένα, υπολογίζει το σφάλμα τους

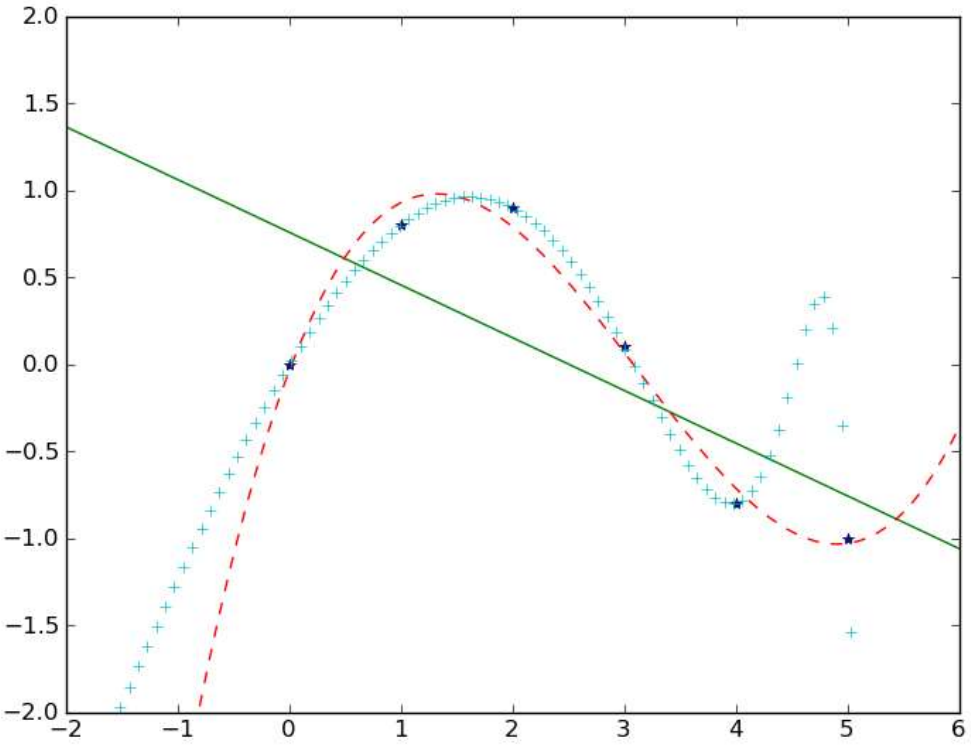
$$E_2(f) = (\frac{1}{n}\sum_{k=1}^{m}|f(x_k) - y_k|^2)^{1/2}$$

και παράγει την γραφική παράσταση δεδομένων και πολυωνύμων.

```
import matplotlib.pyplot as plt
import numpy as np
```

```
from math import sqrt
x = np.array([0.0, 1.0, 2.0, 3.0,  4.0,  5.0])
y = np.array([0.0, 0.8, 0.9, 0.1, -0.8, -1.0])
n=len(x)
xx= np.linspace(-2,6,100)
c1=np.polyfit(x,y,1)
ny1=np.polyval(c1,xx)
c2=np.polyfit(x,y,3)
ny2=np.polyval(c2,xx)
c3=np.polyfit(x,y,30)
ny3=np.polyval(c3,xx)
e1=sqrt( sum(abs(np.polyval(c1,x)-y)**2 )/n )
e2=sqrt( sum(abs(np.polyval(c2,x)-y)**2 )/n )
e3=sqrt( sum(abs(np.polyval(c3,x)-y)**2 )/n )
print e1, e2, e3
plt.plot(x, y, '*', xx, ny1, '-', xx, ny2, '--',xx, ny3, '+')
plt.ylim (-2, 2)
plt.show()
```

το σφάλμα για κάθε πολυώνυμο είναι:
0.499142121187 0.297369419121 2.44818757089e-15
Εξηγείστε το αποτέλεσμα για το πολώνυμο 30ου βαθμού.



7. Ασκήσεις

Άσκηση 1: Ένα απλό πρόβλημα ελαχίστων τετραγώνων (ΕΤ)

Ένας πλανήτης ακολουθεί ελλειπτική τροχιά, που μπορεί να παρασταθεί στις καρτεσιανές συντεταγμένες (x,y) από την εξίσωση: $ay^2 + bxy + cx + dy + e = x^2$

Χρησιμοποιείστε την MATLAB για να προσδιορίσετε τις παραμέτρους a, b, c, d, e λύνοντας το σύστημα ΕΤ, λαμβάνοντας υπόψη τις παρακάτω παρατηρήσεις της θέσης του πλανήτη:

```
x = [ 1.02 0.95 0.87 0.77 0.67 0.56 0.44 0.30 0.16 0.01 ]';
y = [ 0.39 0.32 0.27 0.22 0.18 0.15 0.13 0.12 0.13 0.15 ]';
```

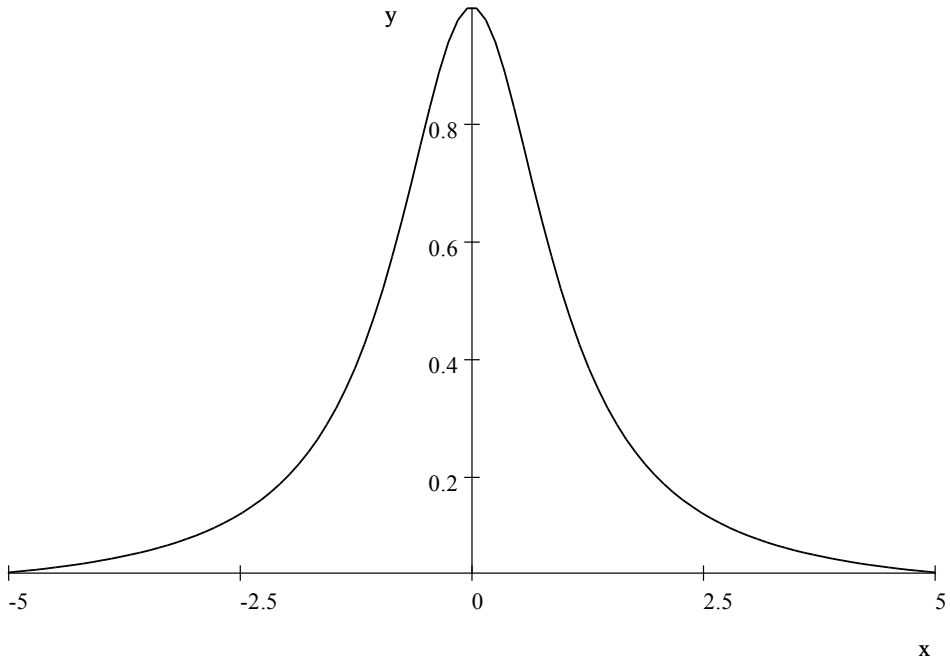
Κάντε την γραφική παράσταση της τροχιάς στα δεδομένα σημεία στο (x, y) επίπεδο.

Το ΕΤ πρόβλημα είναι ασταθές. Για να δείτε την επίδραση που έχει η αστάθεια του στην λύση, μεταβάλετε τα δεδομένα ελαφρώς προσθέτοντας σε κάθε συντεταγμένη του κάθε σημείου ένα τυχαίο αριθμό με ομαλή κατανομή στο διάστημα [-0.005, 0.005] και υπολογίστε την λύση του ΕΤ με τα διαταραχθέντα σημεία.

Ποια είναι η ακρίβεια που παίρνετε με τα διαταραχθέντα σημεία; Ποιος είναι ο αριθμός κατάστασης του συστήματος;

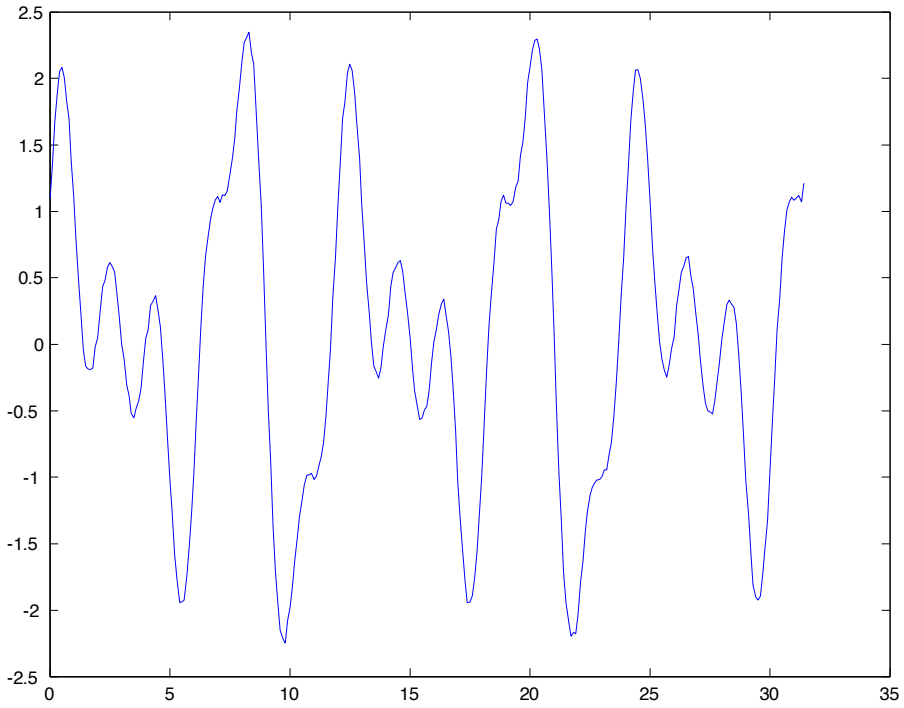
Άσκηση 2: Προσέγγιση ελαχίστων τετραγώνων και λύση υποπροσδιορισμένων συστημάτων

Θεωρήστε τη συνάρτηση $f(x) = \frac{1}{1+x^2}$



στο διάστημα [-5,5] . Στο παραπάνω γράφημα απεικονίζεται η συνάρτηση $f(x)$ στην περιοχή του πεδίου ορισμού της.

Επιπλέον, θεωρήστε τα δεδομένα στο αρχείο data.mat των οποίων το γράφημα είναι



Πρόβλημα 1: Γράψτε ένα MATLAB πρόγραμμα για να βρείτε τα πολυώνυμα βαθμών $n = 3, 7, 15, 21$ που προσεγγίζουν την $f(x)$ λαμβάνοντας υπόψη 60 σημεία της που αντιστοιχούν σε 60 ισαπέχοντα x σημεία στο διάστημα $[-5, 5]$.

- Χρησιμοποιείστε τις κανονικές (normal) εξισώσεις και εφαρμόστε τις QR και SVD μεθόδους για την λύση τους.
- Σχεδιάστε τα πολυώνυμα από κάθε μέθοδο και τα δεδομένα στην ίδια γραφική παράσταση.
- Υπολογίστε και σχεδιάστε τα σφάλματα για κάθε περίπτωση.
- Εξηγήστε τα αποτελέσματα. Βλέπετε μια συγκεκριμένη συμπεριφορά?

Πρόβλημα 2:

- Επαναλάβετε τα βήματα του προβλήματος 1 αλλά για τα δεδομένα στη $2^{η}$ εικόνα.
- Θεωρήστε την προσέγγιση $a\cos(\pi t/3) + b\sin(\pi t/3) + c\sin(\pi t)$ και προσδιορίστε την για το ίδιο σύνολο δεδομένων.
- Εξηγήστε τα αποτελέσματά σας.

Πρόβλημα 3:

Σε όλες τις προσεγγίσεις που υπολογίσατε, υπολογίστε και τη νόρμα του σφάλματος και την Ευκλείδεια νόρμα των υπολοίπων (των σημειακών δηλαδή σφαλμάτων) για κάθε γραμμικό σύστημα που χρησιμοποιήσατε.

Παρατηρήσεις: Μπορεί να βρείτε χρήσιμο το παρακάτω πρόγραμμα σε Matlab

```
%Polynomial least square approximation of the function
%y=exp(sin(4*t))defined in the domain [0,1]
% the user defined parameters of this program are:
% a) m - the number of data points chosen in the interval
[0,1]
% b) n - the degree of the polynomial approximation
% c) mval - the number of points at which the polynomial
approximation is
% computed
% programmer: Elias Houstis
m=30;n=11;mval=60; % parameter definition
t=(0:m-1)'/(m-1); % set a partition of [0,1] with m
elements in array t
y=exp(sin(4*t)); % the values of approximate function at
the partitioning points.
hold off
plot(t,y,'g*') % plots the known data points
hold on
A=[]; %defines Van der Monde matrix
for i=1:n
 A=[A t.^(i-1)]; %generates the Van der Monde matrix
end
a=A'*A; %forms the normal equations
b=A'*y;
% computes the conditional numbers and residuals of the
normal and
% overdetermined systems
cond(a)
cond(A)
%Solve the overdetermined system using Lu and QR
factorization methods.
%Appropriate routines are selected by MATLAB and computes
the norms of the
%residuals
x=a\b;
r=a*x-b;
norm(r)
xqr=A\y;
rqr=A*xqr-y;
norm(rqr)
%permutes the unknown vector according to the requirements
of the polval
```

```
%routine or function
for i=1:n
 p(i)=x(n-i+1);
end
% computes the polynimial at mval points
tt=(0:mval-1)'/(mval-1);
yy=polyval(p,tt);
% plots the polynomial in the same graph with the data
points
plot(tt,yy,'r-')
```

5. Αναφορές

1. http://dmpeli.mcmaster.ca/Matlab/Math1J03/LectureNotes
2. https://sites.google.com/a/uni-konstanz.de/na09/Home/
3. Αριθμητικές Μέθοδοι στην Επιστήμη και Μηχανική, C. Pozrikidis, Εκδόσεις Τζιόλα, 2006
4. Numerical Methods in Engineering with Python, Jaan Kiusalaas, Cambridge University Press, 2005.
5. Numerical Methods for Engineers, with Software and Programming Applications, S.C.Chapra and R.P. Canale, Mc Graw Hill, 2002.
6. Numerical Methods, Software, and Analysis, J.R. Rice, Mc Graw Hill 1983.
7. Applied Numerical Methods, B.C. Carnahan, H.A. Luther, J. O. Wilkes, Krieger Publishing company, 1990.