

# Εργασία #3

(100 μονάδες)

Ημερομηνία Παράδοσης 22/4/2010

## Μη Γραμμικές εξισώσεις και η μέθοδος Newton

Μέχρι τώρα στο μάθημα, μελετήσαμε τεχνικές για την επίλυση γραμμικών εξισώσεων. Εντούτοις, η αλήθεια είναι πως στον πραγματικό κόσμο, η φύση, δεν είναι γραμμική: τρανζίστορ στα δίκτυα, μετατόπιση στην μηχανική, η κίνηση των υγρών, και στοιχειώδη μόρια, και πολλά ακόμη φαινόμενα. Αλλά μην ανησυχείτε, δεν σπαταλήσατε τον χρόνο σας. Οι περισσότερες μέθοδοι επίλυσης μη-γραμμικών εξισώσεων είναι επαναληπτικές και λύνουν ένα γραμμικό πρόβλημα σε κάθε επανάληψη. Η εργασία αυτή αφορά την πιο σημαντική από αυτές τις μεθόδους – την μέθοδο Newton.

### Περιεχόμενα

- [1. Μη-γραμμική συνάρτηση με μία μεταβλητή](#)
- [2. Σύστημα μη-γραμμικών εξισώσεων](#)

### 1. Μη-γραμμική συνάρτηση με μία μεταβλητή

Θα θέλαμε να δείξουμε την μέθοδο Newton επιλύοντας μια γραμμική εξίσωση μίας μεταβλητής:

$$5x = 10 \quad \text{or} \quad f(x) = 5x - 10 = 0$$

Βεβαίως μπορούμε να λύσουμε την εξίσωση σε ένα βήμα, αλλά ας δείξουμε πως η Newton την επιλύει σε δύο βήματα:

1. Μαντέψτε ποια μπορεί να είναι η λύση μας, π.χ.,  $x_0=0$ , που είναι φυσικά λάθος:  $f(0)=-10$ .
2. Αναπροσαρμόστε την πρώτη μαντεψιά χρησιμοποιώντας την πρώτη παράγωγο της συνάρτησης:

$$f'(x_0) = \frac{\Delta f}{\Delta x} = \frac{f(x_1) - f(x_0)}{x_1 - x_0},$$

Όπου  $x_1$  είναι μια αυθαίρετη τιμή. Θα θέλαμε το  $x_1$  να είναι η λύση, δηλαδή,  $f(x_1)=0$ . Τότε:

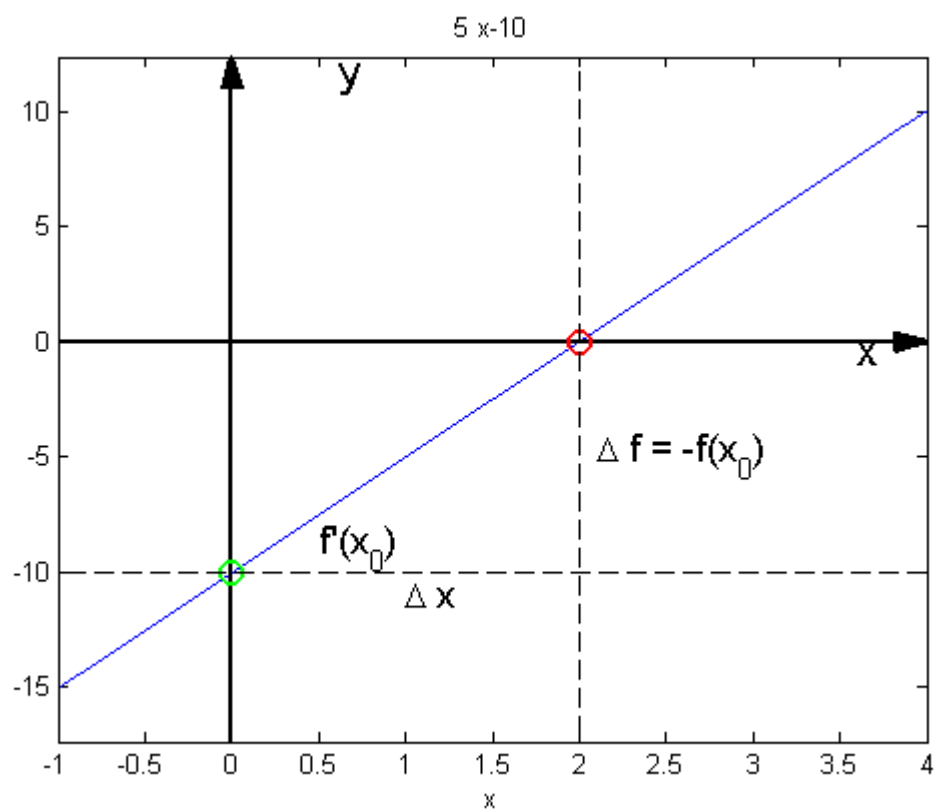
$$f'(x_0)\Delta x = -f(x_0), \quad (1)$$

$$x_1 = x_0 + \Delta x. \quad (2)$$

Αντικαθιστώντας τις τιμές των  $f'(x_0)$ ,  $x_0$  και  $f(x_0)$  στην αντίστοιχη συνάρτηση γυρνάμε πίσω στην αρχική εξίσωση:

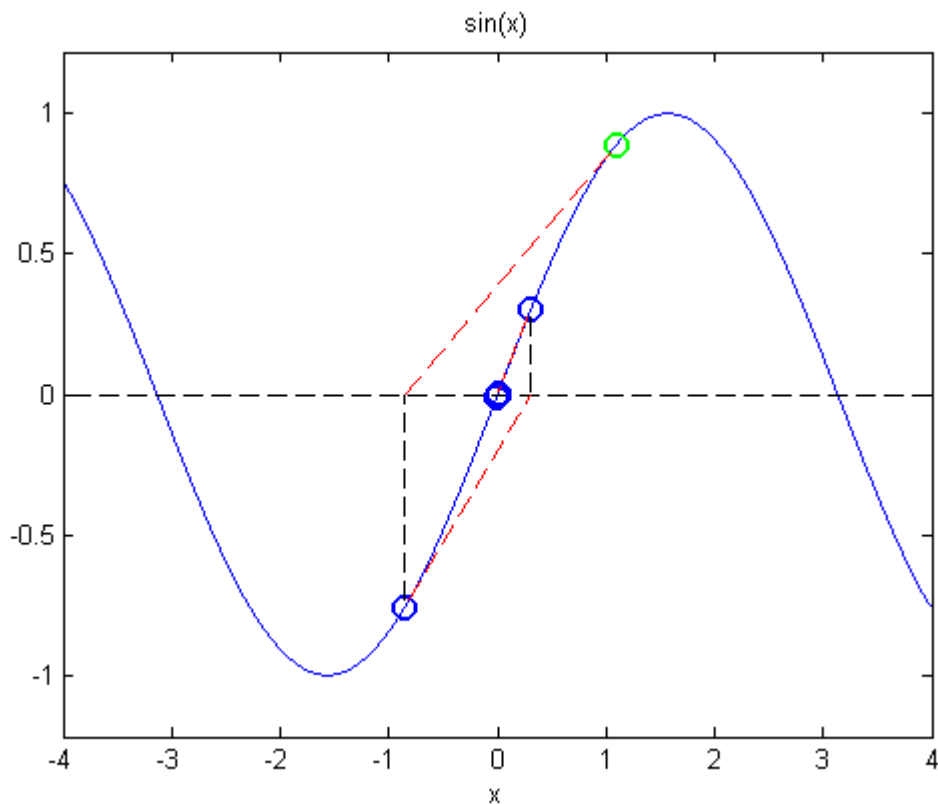
$$f'(x_0)(x_1 - x_0) = 5x_1 = -(-10).$$

Η παρακάτω εικόνα δίνει μια γεωμετρική ερμηνεία αυτής της ιδέας:



Η ίδια ιδέα εφαρμόζεται και στην μη-γραμμική περίπτωση, π.χ., για να βρούμε το μηδέν σε ημιτονική συνάρτηση,  $f(x) = \sin(x)$ . Παρόλα αυτά, ο αλγόριθμος δεν σταματάει στο  $x_1$ , αλλά επαναλαμβάνεται, χρησιμοποιώντας τις Εξ. (1) και (2) που παράγουν μια (ενδεχομένως συγκλίνουσα) σειρά  $x_0, x_1, x_2, \dots, x_k$  με  $f(x_k) = 0$ . Η παρακάτω εικόνα είναι καλύτερη από τις παραπάνω 40 λέξεις :

# of iterations: 5



### Άσκηση 1: `nlnewton()` και τετραγωνική σύγκλιση

Η μέθοδος υλοποιείται στην συνάρτηση `nlnewton()` που είναι διαθέσιμη στο `eclass`. Χρειάζεται 3 υποχρεωτικά ορίσματα: δύο υπολογίζουν τα  $f(x)$  και  $f'(x)$  και την αρχική  $x_0$  που μαντέψαμε.

- Η βοήθεια για την συνάρτηση `nlnewton()` περιλαμβάνει ένα παράδειγμα για την αξιολόγηση της έκφρασης `sqrt(2)`. Τρέξτε αυτό το παράδειγμα. Πόσα σωστά ψηφία παίρνετε;
- Το 4<sup>ο</sup> προαιρετικό όρισμα της `nlnewton()` είναι για μια συνάρτηση που ορίζει ο χρήστης και καλείται σε κάθε επανάληψη. Η συνάρτηση παίρνει 4 παραμέτρους εισόδου: αριθμός επαναλήψεων, τωρινό  $x_0$ , υπολογισμένη  $dx$  και το  $f(x)$  για την μη-γραμμική συνάρτηση. Ένα παράδειγμα της συνάρτησης, `IterFcn.m` είναι διαθέσιμο στο `eclass`. Τρέξτε τον παρακάτω κώδικα και παρατηρήστε την **τετραγωνική σύγκλιση**:

```
format long
format compact
f=@(x) x.^2-2; J=@(x) 2*x; x0=1; x=nlnewton(f,J,x0,@IterFcn);
```

Ο ΚΩΔΙΚΑΣΣΑΣΕΔΩ:

0	1	-1
1.0000000000000000	1.5000000000000000	0.2500000000000000
2.0000000000000000	1.4166666666666667	0.0069444444444445
3.0000000000000000	1.414215686274510	0.000006007304883
4.0000000000000000	1.414213562374690	0.0000000000004511

5.0000000000000000 1.414213562373095 0.0000000000000000  
# of iterations: 5

## Άσκηση 2: οπτικοποίηση σύγκλισης

Τροποποιείτε την συνάρτηση `IterFcn.m`, έτσι ώστε

- Στην 0 επανάληψη, να δημιουργεί ένα διάγραμμα της δοσμένης συνάρτησης και ζωγραφίζει την αρχική μαντεψιά ( $x_0$ ,  $f(x_0)$ ). Υπόδειξη: χρησιμοποιείτε την συνάρτηση του `ezplot(f, [x1 x2])` για να κάνετε το γράφημα της  $f(x)$  στο διάστημα  $[x1 \ x2]$ ;
- Στις επόμενες επαναλήψεις, αναπαριστά τις εφαπτόμενες γραμμές του διαγράμματος της  $f(x)$  στο  $x_0$  που διασταυρώνονται με τον άξονα  $x$  και την νέα τιμή ( $x_1$ ,  $f(x_1)$ ). Το τελικό διάγραμμα θα πρέπει να είναι ίδιο με το παραπάνω.

Με την νέα `IterFcn()`, παρατηρείστε την διαδικασία επίλυσης των παρακάτω συναρτήσεων:

- $f(x) = x^2 - 2$  στο διάστημα  $[0 \ 10]$  για αρχικές τιμές  $x_0 = 1, .5, .1, 0$ ;
- $f(x) = \sin(x)$  στο διάστημα  $[-4 \ 4]$ . Δοκιμάστε τις πέντε αρχικές τιμές:  $x_0 = 1.1 : .1 : 1.5$ . Ο αλγόριθμος συγκλίνει πάντα στο κοντινότερο μηδέν;  
 $f(x) = x^3 - 3x^2 + 5$  στο διάστημα  $[-4 \ 4]$  για  $x_0 = -3 : .1 : 3$  iteration -use the MATLAB rapid code) buttons to iterate through  $x_0$  | s) . Η σύγκλιση γίνεται πάντα με ομαλό τρόπο; Ποιο είναι το πρόβλημα με το σημείο  $|0|=1$ ?
- $f(x) = \text{sign}(x-2) \cdot \sqrt{\text{abs}(x-2)}$  στο διάστημα  $[0 \ 4]$  με  $x_0 = 1$ . Η μέθοδος συγκλίνει;... αποκλίνει; Γιατί;

Ο ΚΩΔΙΚΑΣΣΑΣΕΔΩ:

## Άσκηση 3: μέθοδος secant

Παρόλο που η μέθοδος έχει αξιοσημείωτες ιδιότητες σύγκλισης, έχει ένα μεγάλο μειονέκτημα: μπορεί να είναι αρκετά ακριβή, καθώς ο υπολογισμός της παραγώγου της συνάρτησης σε κάθε βήμα μπορεί να είναι δύσκολο εάν όχι αδύνατο. Είναι πάντα εύκολο να προσεγγίσουμε την παράγωγο αριθμητικά με πεπερασμένες διαφορές χρησιμοποιώντας ένα μικρό βήμα  $h$ :

$$f'(x) = \frac{f(x+h) - f(x)}{h}$$

Εντούτοις, αυτή η προσέγγιση είναι ακόμη ακριβή γιατί απαιτεί τον πρόσθετο υπολογισμό της συνάρτησης,  $f(x+h)$ . Η μέθοδος αναπαριστά μια φθηνότερη (αλλά λιγότερο καλή) εναλλακτική που αντικαθιστά την παράγωγο με πεπερασμένη διαφορά που βασίζεται στις δύο πιο πρόσφατες επαναλήψεις που είναι διαθέσιμες ούτως ή άλλως:

$$f'(x) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

- Υλοποιείτε την μέθοδο στην συνάρτηση `nlsecant()` τροποποιώντας τον κώδικα `nlnewton()`. Η συνάρτησή σας θα απαιτεί μόνο 2 υποχρεωτικά ορίσματα: την συνάρτηση και το σημείο εκκίνησης.

- Εφαρμόστε την `nlsecant()` στην συνάρτηση της προηγούμενης άσκησης. Συγκρίνετε η σύγκλιση με αυτή της `nlnewton()` ?

Ο ΚΩΔΙΚΑΣ ΣΑΣ ΕΔΩ:

#### Άσκηση 4: MATLAB `fzero()`

Η συνάρτηση του `fzero()` βρίσκει το μηδέν μιας κλιμακωτής συνάρτησης με έναν συνδυασμό των μεθόδων *bisection* και *inverse quadratic interpolation*. Δε θα δούμε τις λεπτομέρειες των δύο τελευταίων προσεγγίσεων, αλλά παρατηρήστε πως αυτές μπορεί αν είναι σταθείς:

- use `fzero(f,x0)` για να βρείτε το `sqrt(1)` με σημεία εκκίνησης `x0=10` και `x0=11`.

YOUR CODE HERE:

## 2. Σύστημα μη-γραμμικών εξισώσεων

Εκτός από την απλότητα, η ομορφιά της μεθόδου έγκειται στο γεγονός της εύκολης γενίκευσης από την κλιμακωτή συνάρτηση μιας μεταβλητής σε ένα σύστημα  $n$  εξισώσεων με  $n$  μεταβλητές. Οι Εξ. (1) και (2) δεν αλλάζουν: τα  $\mathbf{x}$  και  $\mathbf{f}(\mathbf{x})$  γίνονται  $n$  - διανύσματα:

$$\mathbf{x} = [x_1, x_2, \dots, x_n]^T,$$

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_n(\mathbf{x})]^T,$$

και η παράγωγος  $\mathbf{f}'(\mathbf{x})$  αντικαθίσταται με το πολυδιάστατο ανάλογο - ένα  $n$ -επί- $n$  **Jacobian πίνακα**  $\mathbf{J}(\mathbf{x})$ :

$$\mathbf{J}(\mathbf{x})(i, j) = \frac{\partial f_i(\mathbf{x})}{\partial x_j}.$$

Η Εξ (1) γίνεται εξίσωση πινάκων:

$$\mathbf{J}(\mathbf{x}) \cdot \Delta \mathbf{x} = -\mathbf{f}(\mathbf{x}_0),$$

Όπου η Εξ. (2) παραμένει άθικτη.

#### Άσκηση 5: πολυδιάστατη μέθοδος Newton

- Τροποποιείτε την συνάρτηση `nlnewton.m` για να βρίσκει το μηδέν μιας συνάρτησης με διανύσματα. Υπόδειξη: χρειάζεται να τροποποιήσετε δύο γραμμές μόνο.
- Η βοήθεια του `nlnewton()` περιλαμβάνει ένα παράδειγμα εφαρμογής της μεθόδου για σύστημα μη-γραμμικών εξισώσεων 2-επί-2. Ελέγξτε τις συναρτήσεις σας σε αυτό το παράδειγμα. Δοκιμάστε διαφορετικά αρχικά διανύσματα, π.χ. `x0=[1 2]'`.
- Τι κάνει η μέθοδος για:

$$x_1^3 - x_2 = 0,$$

$$x_2^3 - x_1 = 0?$$

Ο ΚΩΔΙΚΑΣ ΣΑΣ ΕΔΩ:

### Άσκηση 6: μέθοδος Newton

Καθένα από τα παρακάτω συστήματα μη-γραμμικών εξισώσεων μπορεί να παρουσιάζουν κάποια δυσκολία στον υπολογισμό της λύσης. Χρησιμοποιήστε την συνάρτηση `nlnewton()` για να επιλύσετε τα συστήματα για το δοσμένο σημείο εκκίνησης. Ο μέγιστος αριθμός των επαναλήψεων να είναι 100.

Σε μερικές περιπτώσεις, η δική σας –συνάρτηση μπορεί να αποτυγχάνει και να μην συγκλίνει ή μπορεί να συγκλίνει σε ένα άλλο σημείο και όχι στην λύση. Όταν αυτό συμβαίνει, προσπαθήστε να εξηγήσετε τον λόγο αυτής της συμπεριφοράς.

a)

$$\begin{aligned} x_1 + x_2(x_2(5 - x_2) - 2) &= 13, \\ x_1 + x_2(x_2(1 + x_2) - 14) &= 29, \quad \mathbf{x}_0 = [15, -2]^T. \end{aligned}$$

b)

$$\begin{aligned} x_1^2 + x_2^2 + x_3^2 &= 5, \\ x_1 + x_2 &= 1, \\ x_1 + x_3 &= 3, \quad \mathbf{x}_0 = \left[ \frac{1 + \sqrt{3}}{2}, \frac{1 - \sqrt{3}}{2}, \sqrt{3} \right]^T. \end{aligned}$$

c)

$$\begin{aligned} x_1 + 10x_2 &= 0, \\ \sqrt{5}(x_3 - x_4) &= 0, \\ (x_2 - x_3)^2 &= 0, \\ \sqrt{10}(x_1 - x_4)^2 &= 0, \quad \mathbf{x}_0 = [1, 2, 1, 1]^T. \end{aligned}$$

d)

$$\begin{aligned} x_1 &= 0, \\ \frac{10x_1}{x_1 + 0.1} + 2x_2^2 &= 0, \quad \mathbf{x}_0 = [1.8, 0]^T. \end{aligned}$$

e)

$$\begin{aligned} 10^4 x_1 x_2 &= 1, \\ e^{-x_1} + e^{-x_2} &= 1.0001, \quad \mathbf{x}_0 = [0, 1]^T. \end{aligned}$$

### Άσκηση 7: Newton vs. Breuden

Στο τέλος της άσκησης θα βρείτε το πρόγραμμα του που υλοποιεί την **μέθοδο Breuden**. Είναι μια πολύ-διάστατη παραλλαγή της μεθόδου

- Υλοποιείστε την μέθοδο Breuden στην συνάρτηση `nlbreuden()`. Χρησιμοποιείτε τον ταυτοτικό πίνακα σαν αρχική προσέγγιση του πίνακα.

Θεωρείστε το παρακάτω σύστημα μη-γραμμικών εξισώσεων:

$$(x_1 + 3)(x_2^3 - 7) + 18 = 0,$$

$$\sin(x_2 e^{x_1} - 1) = 0.$$

- Επιλύστε το σύστημα χρησιμοποιώντας την μέθοδο με σημείο εκκίνησης  $\mathbf{x}_0 = [-1, 1.4]^T$ .

Επιλύστε το σύστημα χρησιμοποιώντας την μέθοδο με το ίδιο σημείο εκκίνησης.

Συγκρίνετε τους ρυθμούς σύγκλισης των δύο μεθόδων υπολογίζοντας το σφάλμα σε κάθε επανάληψη, δεδομένης της ακριβής λύσης  $x = [0, 1]'$ . Παράγεται ένα γράφημα της 2-νόρμας του σφάλματος ως προς τον αριθμό των επαναλήψεων της κάθε μεθόδου.

Πόσες επαναλήψεις χρειάζεται κάθε μέθοδος για να επιτύχει ακρίβεια ( );

```
function y = broyden ( F, J, x0, TOL, Nmax )

%BROYDEN      solve the system of nonlinear equations F(x) = 0 using
%              Broyden's method
%
%      calling sequences:
%      y = broyden ( F, J, x0, TOL, Nmax )
%      broyden ( F, J, x0, TOL, Nmax )
%
%      inputs:
%      F        vector-valued function of a vector argument which
%                defines the system of equations to be solved
%      J        matrix-valued function which computes the Jacobian
%                associated with the function F
%      x0       vector containing initial guess for solution of
%                nonlinear system
%      TOL      convergence tolerance - applied to maximum norm of
%                difference between successive approximations
%      NMax     maximum number of iterations to be performed
%
%      output:
%      y        approximate solution of nonlinear system
%
%      NOTE:
%      if BROYDEN is called with no output arguments, each
%      approximation to the solution is displayed
%
%      if the maximum number of iterations is exceeded, a message
%      to this effect will be displayed and the current approximation
%      will be returned in the output value
%

Fold = feval(F,x0)';
Jold = feval(J,x0);
A0 = inv ( Jold );
dx = -A0 * Fold;
x0 = x0 + dx;
if ( nargout == 0 )
    disp ( x0' )
end

for i = 2 : Nmax
    Fnew = feval(F,x0)';
    dy = Fnew - Fold;
    u = A0 * dy;
    v = dx' * A0;
    denom = dx' * u;
    A0 = A0 + ( dx - u ) * v / denom;
    dx = -A0 * Fnew;
```

```

x0 = x0 + dx;

if ( nargout == 0 )
    disp ( x0' )
end

if ( max(abs(dx)) < TOL )
    if ( nargout == 1 )
        y = x0;
    end
    return
else
    Fold = Fnew;
end

end

disp('broyden error: Maximum number of iterations exceeded');
if ( nargout == 1 ) y = x0; end;

```