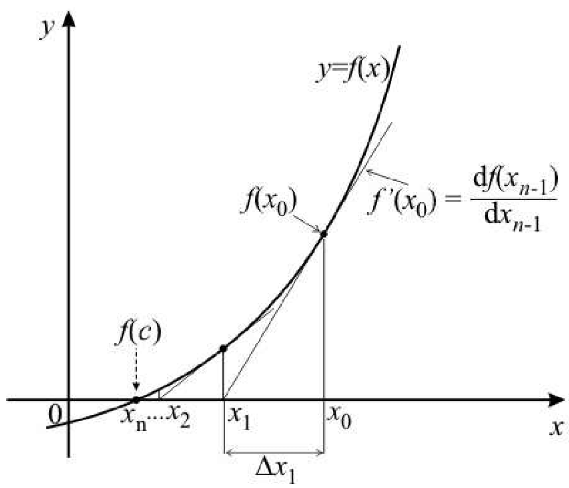
1η Εργασία Στο Μάθημα Της Αριθμητικής Ανάλυσης

Ονοματεπώνυμο : Παπαδόπουλος Αναστάσιος

ΑΕΜ : 3654

Διδάσκων : Τέφας Αναστάσιος

6 Δεκεμβρίου 2020



ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Περιεχόμενα

1 Εισαγωγή

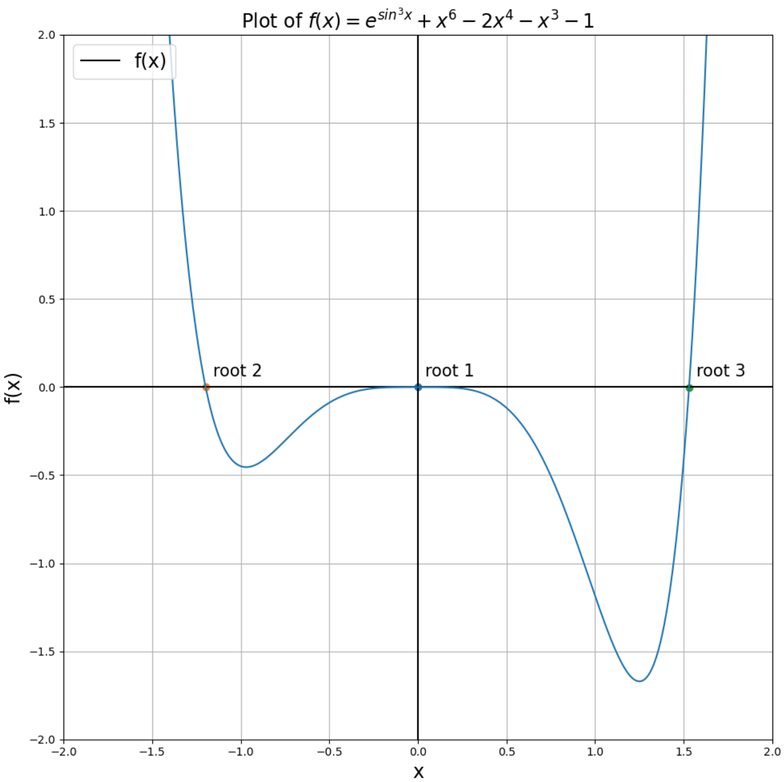
Η εργασία αυτή αναπτύχθηκε κατά την παρακολούθηση του μάθηματος της Αριθμητικής Ανάλυσης του 3ου εξαμήνου του τμήματος Πληροφορικής ΑΠΘ. Περιλαμβάνει υλοποιήσεις και πειραματισμούς με αλγορίθμους Αριθμητικής Ανάλυσης που αναφέρονται σε θεματικές ενότητες όπως της επίλυσης μη γραμμικών εξισώσεων (**Άσκηση 1 και 2**) για παράδειγμα η μέθοδος της διχοτόμησης, της επίλυσης γραμμικών συστημάτων (**Άσκηση 3**) για παράδειγμα η μέθοδος της $\mathbf{PA} = \mathbf{LU}$ αποσύνθεσης και τέλος της ιδιοανάλυσης (**Άσκηση 4**). Οι αλγόριθμοι αυτοί καθώς και οι γραφικές παραστάσεις που περιέχονται στο παρόν έγγραφο έχουν υλοποιηθεί με την γλώσσα προγραμματισμού **Python** με χρήση των βιβλιοθηκών **NumPy** και **Matplotlib**. Στα παραδοτέα περιλαμβάνεται το παρόν **pdf** αρχείο καθώς και το αντίστοιχο **.tex** και για κάθε μία άσκηση υπάρχει ένας υποφάκελος όπου περίεχει τα αντίστοιχα **.py** αρχεία με τον πηγαίο κώδικα.

2 Άσκηση 1

Στην **1η άσκηση** μας δίνεται αρχικά η συνάρτηση

$$f(x) = e^{sin^3 x} + x^6 - 2x^4 - x^3 - 1$$

και ζητείται η γραφική παράσταση της στο διάστημα [-2,2].



Σχήμα 1: Γραφική παράσταση της συνάρτησης $\mathbf{f(x)}$

Από την οποία παρατηρούμε στο *Σχήμα 1* ότι η συνάρτηση έχει 3 ρίζες.Στην συνέχεια ζητείται να υπολογισθούν αυτές οι ρίζες με ακρίβεια 5ου δεκαδικού ψηφίου χρησιμοποιώντας την μέθοδο της διχοτόμησης, την μέθοδο Newton-Raphson και τέλος την μέθοδος της τέμνουσας.

2.1 Μέθοδος της διχοτόμησης

Η μέθοδος της διχοτόμησης γενικά προϋποθέτει την ύπαρξη ενός διαστήματος **[a,b]** στο οποίο η συνάρτηση έχει τουλάχιστον μία ρίζα. Για να βρούμε ένα τέτοιο διάστημα χρησιμοποιούμε το θεώρημα **Bolzano** σε συνδυασμό με την γραφική παράσταση της **f(x)**. Όπως παρατηρούμε στο *Σχήμα 1* η **f(x)** έχει μία ρίζα στο διάστημα **[-1.5,-1.0]**. Εξάλλου, στο διάστημα **[-1.5,-1.0]** η **f(x)** είναι συνεχής με

$$\mathbf{f(-1.5)} * \mathbf{f(-1.0)} < 0$$

συνεπώς σύμφωνα με το θέωρημα **Bolzano** έχει μία τουλάχιστον ρίζα στο διάστημα αυτό. Για να βρούμε την ρίζα χρησιμοποιούμε την συνάρτηση ***bisection*** από το αρχείο ***bisection.py*** με ορίσματα την συνάρτηση **f(x), a = -1.5, b = -1.0** , ενώ το **default** όρισμα **eps** έχει την τιμή που χρειάζεται για ακρίβεια **5** δεκαδικών ψηφίων. Η συνάρτηση επιστρέφει την ρίζα της **f(x)** στο διάστημα **(a,b)** και τον αριθμό των επαναλήψεων που χρειάστηκαν για να βρέθει η ρίζα με σφάλμα λιγότερο από **eps**.

```
root, iterations_num = bisection(g, -1.5, -1.0)
```

Σχήμα 2: Παράδειγμα κλήσης της συνάρτησης ***bisection***

Μετά την κλήση της συνάρτησης ***bisection*** τα αποτελέσματα είναι τα εξής:

```
Root = -1.19762
Iterations = 17
```

Σχήμα 3: Αποτελέσματα κλήσης της συνάρτησης ***bisection*** στο διάστημα **[-1.5,-1.0]**.

Όπου παρατηρούμε ότι η ρίζα της **f(x)** στο διάστημα **[-1.5,-1.0]** με ακρίβεια 5 δεκαδικών ψηφίων είναι η **-1.19762** καθώς και ότι η μέθοδος της διχοτόμησης χρειάστηκε **17** επαναλήψεις για να επιτύχει την επιθυμητή ακρίβεια. Στην συνέχεια παρατηρούμε στο *Σχήμα 1* ότι η **f(x)** έχει μία ακόμα ρίζα στο διάστημα **[1.25,1.75]**. Πέρα από την γραφική παράσταση, με παρόμοιο τρόπο χρησιμοποιώντας το θεώρημα **Bolzano** μπορούμε να αποδείξουμε ότι η **f(x)** όντως έχει μία τουλάχιστον ρίζα στο διάστημα **[1.25,1.75]**. Επομένως καλούμε πάλι την συνάρτηση ***bisection*** αυτή την φορά όμως με ορίσματα **f(x), a = 1.25, b = 1.75** .

```
root, iterations_num = bisection(g, 1.25, 1.75)
```

Σχήμα 4: Παράδειγμα κλήσης της συνάρτησης ***bisection***

Από την οποία παίρνουμε τα εξής αποτελέσματα:

```
Root = 1.53013
Iterations = 17
```

Σχήμα 5: Αποτελέσματα κλήσης της συνάρτησης ***bisection*** στο διάστημα **[1.25,1.75]**.

Όπου παρατηρούμε ότι η ρίζα της **f(x)** στο διάστημα **[1.25,1.75]** με ακρίβεια 5 δεκαδικών ψηφίων είναι η **1.53013** καθώς και ότι η μέθοδος της διχοτόμησης χρειάστηκε **17** επαναλήψεις για να επιτύχει την επιθυμητή ακρίβεια. Τέλος, μπορούμε εύκολα να αποδείξουμε ότι η τρίτη και τελευταία ρίζα της **f(x)** είναι το **0** καθώς **f(0) = 0**. Στην περίπτωση μας όμως δεν υπάρχει κανένα διάστημα που ταυτόχρονα να περιέχει το **0** και το θεώρημα **Bolzano** να εγγυάται την ύπαρξη τουλάχιστον μιας ρίζας καθώς δεν υπάρχουν **x1,x2** με

$$\mathbf{x1} \neq \mathbf{x2}, \quad \mathbf{x1}, \mathbf{x2} \in \textbf{[-2,2]}$$

και

$$\mathbf{f(x1)} * \mathbf{f(x2)} < 0$$

και

$$\mathbf{0} \in \mathbf{[x1,x2]}$$

Συνεπώς η κλασσική μέθοδος της διχοτόμησης δεν εγγυάται σύγκλιση σε αυτή την περίπτωση καθώς και δεν μπορεί να χρησιμοποιηθεί για την εύρεση της ρίζας $\mathbf{x} = \mathbf{0}$ αφού προϋποθέτει την ύπαρξη ενός διαστήματος στο οποίο να ισχύει το θεώρημα **Bolzano** για την $\mathbf{f(x)}$. Αν προσπαθήσουμε να καλέσουμε την συνάρτηση ***bisection*** για ένα διάστημα που δεν ισχύει το θεώρημα **Bolzano** τα αποτελέσματα είναι μη προβλέψιμα και ανάλογα την περίπτωση και τις τιμές του διαστήματος **[a,b]** η μέθοδος μπορεί να μην συγκλίνει σε σωστή λύση ή να συγκλίνει αλλά όχι με την επιθυμητή ακρίβεια.

```
Root = 0.05000
Iterations = 15
```

Σχήμα 6: Αποτελέσματα κλήσης της συνάρτησης ***bisection*** στο διάστημα **[-0.05,0.05]**.

```
Root = 0.50000
Iterations = 18
```

Σχήμα 7: Αποτελέσματα κλήσης της συνάρτησης ***bisection*** στο διάστημα **[-0.5,0.5]**.

Η συνάρτηση ***bisection*** που έχει υλοποιηθεί στο αρχείο ***bisection.py*** ελέγχει αν στο διάστημα **[a,b]** ισχύει

$$\mathbf{f(x1)} * \mathbf{f(x2)} > 0$$

και αν ναι επιστρέφει την τιμή **nan** ως ρίζα και την τιμή **-1** ως τον αριθμό των επαναλήψεων. Η συνάρτηση έχει τροποποιηθεί επίσης ώστε να ελέγχει αν τα άκρα του αρχικού διαστήματος **[a,b]** αποτελούν ρίζες της **f(x)**. Αν προσπαθήσουμε τώρα να καλέσουμε την συνάρτηση ***bisection*** στην τροποποιημένη μορφή της παρατηρούμε τα εξής:

```
Root = nan
Iterations = -1
```

Σχήμα 8: Αποτελέσματα κλήσης της τροποποιήμενης συνάρτησης ***bisection*** στο διάστημα **[-0.5,0.5]**.

```
Root = 0.00000
Iterations = 0
```

Σχήμα 9: Αποτελέσματα κλήσης της τροποποιήμενης συνάρτησης ***bisection*** στο διάστημα **[0,1]**.

όπου στην πρώτη κλήση δεν ισχύει το θέωρημα **Bolzano** ενώ στην δεύτερη το ένα άκρο του διαστήματος είναι ρίζα της **f(x)**. Τέλος, τονίζεται ότι αν και τα δύο άκρα του διαστήματος είναι ρίζες της **f(x)** επιστρέφεται η τιμή του **a**.

2.2 Μέθοδος Newton-Raphson

Η μέθοδος **Newton-Raphson** αποτελεί μία περίπτωση μεθόδου σταθερού σημείου. Προφανώς προϋποθέτει κι αυτή, όπως και η μέθοδος της διχοτόμησης, την ύπαρξη ενός διαστήματος **[a,b]** στο οποίο η συνάρτηση έχει τουλάχιστον μία ρίζα. Κατ΄ αναλογία με την μέθοδο της διχοτόμησης για να βρούμε και πάλι ένα τέτοιο διάστημα χρησιμοποιούμε το θεώρημα **Bolzano** σε συνδυασμό με την γραφική παράσταση της **f(x)**. Εκτός από την ύπαρξη διαστήματος με τις προαναφερόμενες ιδιότητες, η μέθοδος **Newton-Raphson** απαιτεί η εκάστοτε συνάρτηση **f(x)** να είναι δύο φορές παραγωγίσιμη σε αυτό το διάστημα με

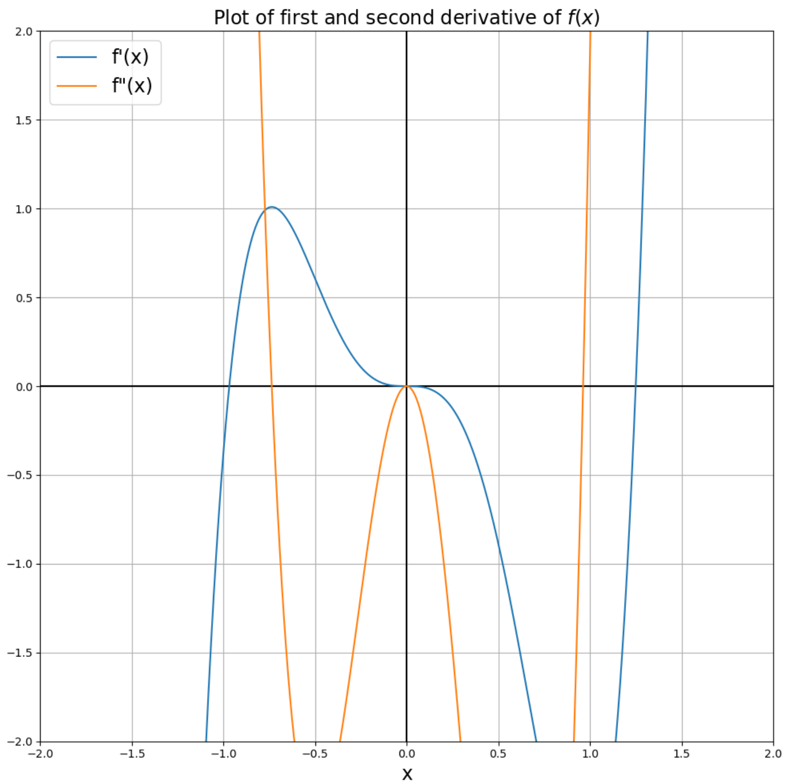
$$f'(x), f''(x) \neq \mathbf{0} \quad \forall \mathrm{x} \in \mathbf{[a,b]}$$

Καθώς και ένα αρχικό σημείο x_0 από το οποίο θα ξεκινήσει η επαναληπτική μέθοδος με την εξής ιδιότητα

$$\mathrm{f}(x_0) * f''(x_0) > \mathbf{0}$$

Όμοια με πριν παρατηρούμε στο *Σχήμα 1* ότι η δοσμένη **f(x)** έχει μία ρίζα στο διάστημα **[-1.5,-1.0]** καθώς και ότι ισχύει το θεώρημα **Bolzano** σε αυτό το διάστημα. Επίσης, παρατηρούμε στο *Σχήμα 10* ότι ισχύει

$$f'(x), f''(x) \neq \mathbf{0} \quad \forall \mathrm{x} \in [\text{-}1.5,\text{-}1.0]$$



Σχήμα 10: Γραφική παράσταση της πρώτης και δεύτερης παραγώγου της συνάρτησης $\mathbf{f(x)}$

Επομένως αρκεί να βρούμε ένα σημείο x_0 με

$$x_0 \in \textbf{[-1.5,-1.0]}$$

τέτοιο ώστε να ικανοποιείται η συνθήκη

$$\mathrm{f}(x_0) * f''(x_0) > \mathbf{0}$$

Ένα τέτοιο σημείο είναι το

$$x_0 = \textbf{-1.4}$$

Για να βρούμε την ρίζα χρησιμοποιούμε την συνάρτηση ***newton_raphson*** από το αρχείο ***newton_raphson.py*** με ορίσματα την συνάρτηση **f(x)**, την παράγωγο της **f'(x)** και το αρχικό σημείο x_0= **-1.4**, το **default** όρισμα **eps** έχει την τιμή που χρειάζεται για ακρίβεια **5** δεκαδικών ψηφίων ενώ το **default** όρισμα **max_iterations** αντιστοιχεί στον μέγιστο αριθμό επαναλήψεων που θα γίνουν σε περίπτωση που έχει δοθεί κάποιο αρχικό σημείο x_0 χωρίς να τηρούνται οι προϋποθέσεις που εγγυώνται σύγκλιση για την μέθοδο **Newton-Raphson**, με τιμή τον αριθμό **50**. Αν πληρούνται όλες οι προϋποθέσεις η συνάρτηση επιστρέφει την ρίζα της **f(x)** στο αντίστοιχο διάστημα **(a,b)** (εδώ στο **(-1.5,-1.0)**), αφού οι προαναφερόμενες προϋποθέσεις εγγυώνται την ύπαρξη **μοναδικής** ρίζας στο διάστημα αυτό, καθώς και τον αριθμό των επαναλήψεων που χρειάστηκαν για να βρέθει η ρίζα με σφάλμα λιγότερο από **eps**.

```
root, iterations_num = newton_raphson(g, gprime, -1.4)
```

Σχήμα 11: Παράδειγμα κλήσης της συνάρτησης ***newton_raphson*** με αρχική τιμή x_0 = **-1.4**.

Μετά την κλήση της συνάρτησης ***newton_raphson*** τα αποτελέσματα είναι τα εξής:

```
Root = -1.19762
Iterations = 4
```

Σχήμα 12: Αποτελέσματα κλήσης της συνάρτησης ***newton_raphson*** με αρχική τιμή x_0 = **-1.4**.

Όπου παρατηρούμε ότι η ρίζα της **f(x)** στο διάστημα **[-1.5,-1.0]** με ακρίβεια 5 δεκαδικών ψηφίων είναι η **-1.19762** καθώς και ότι η μέθοδος **Newton-Raphson** χρειάστηκε **4** επαναλήψεις για να επιτύχει την επιθυμητή ακρίβεια. Στην συνέχεια παρατηρούμε στο *Σχήμα 1* ότι η **f(x)** έχει μία ακόμα ρίζα στο διάστημα **[1.25,1.75]** και πως ισχύει το θέωρημα **Bolzano** σε αυτό το διάστημα, όμως δεν ικανοποιείται η πρώτη συνθήκη οπότε θα χρειαστεί να περιορίσουμε το διάστημα στο **[1.3,1.75]** όπου έχουμε

$$f'(x), f''(x) \neq \mathbf{0} \quad \forall \mathrm{x} \in \mathbf{[1.3,1.75]}$$

Επομένως, το επόμενο βήμα είναι η επιλογή του αρχικού σημείου x_0 έτσι ώστε να ικανοποιείται και η τρίτη συνθήκη. Ένα τέτοιο σημείο είναι το $x_0 = \mathbf{1.7}$. Καλούμε λοιπόν την συνάρτηση ***newton_raphson*** με αρχικό σημείο το $x_0 = \mathbf{1.7}$.

```
root, iterations_num = newton_raphson(g, gprime, 1.7)
```

Σχήμα 13: Παράδειγμα κλήσης της συνάρτησης ***newton_raphson*** με αρχική τιμή $x_0 = \mathbf{1.7}$.

Μετά την κλήση της συνάρτησης ***newton_raphson*** τα αποτελέσματα είναι τα εξής:

```
Root = 1.53013
Iterations = 4
```

Σχήμα 14: Αποτελέσματα κλήσης της συνάρτησης ***newton_raphson*** με αρχική τιμή $x_0 = \mathbf{1.7}$.

Όπου παρατηρούμε ότι η ρίζα της **f(x)** στο διάστημα **[1.3,1.75]** με ακρίβεια 5 δεκαδικών ψηφίων είναι η **1.53013** καθώς και ότι η μέθοδος **Newton-Raphson** χρειάστηκε **4** επαναλήψεις για να επιτύχει την επιθυμητή ακρίβεια. Τέλος, αφού η μέθοδος **Newton-Raphson** προϋποθέτει κι αυτή, όπως και η μέθοδος της διχοτόμησης, την ύπαρξη διαστήματος στο οποίο ισχύει το θεώρημα **Bolzano** για να εγγυάται η σύγκλιση (εφόσον ισχύουν και οι υπόλοιπες προϋποθέσεις) αντιμετωπίζουμε πάλι πρόβλημα με την ρίζα $\mathbf{x} = \mathbf{0}$. Αν προσπαθήσουμε να καλέσουμε την συνάρτηση ***newton_ raphson*** με αρχικό σημείο κάποιο σημείο στο οποίο δεν τηρούνται όλες οι προϋποθέσεις τα αποτελέσματα είναι μη προβλέψιμα και είτε μπορεί να έχουμε σύγκλιση σε λάθος αποτέλεσμα είτε σύγκλιση χωρίς την επιθυμητή ακρίβεια.

x_0	Αποτέλεσμα κλήσης συνάρτησης ***newton_ raphson***	Αριθμός Επαναλήψεων
0.25	0.00005	29
0.5	0.00009	30
-0.5	-0.00005	31
-0.25	-0.00004	29

Σχήμα 15: Αποτέλεσματα κλήσεων της συνάρτησης ***newton_ raphson*** χωρίς να τηρούνται όλες οι προϋποθέσεις

Όπως παρατηρούμε και στο σχήμα 15, καμία κλήση δεν βρίσκει την ρίζα της **f(x)** με την επιθυμητή ακρίβεια. Η συνάρτηση ***newton_ raphson*** που έχει υλοποιηθεί στο αρχείο ***newton_ raphson.py*** έχει τροποποιηθεί έτσι ώστε να ελέγχει αν το αρχικό σημείο x_0 αποτελεί ρίζα της **f(x)**. Έτσι, αν τώρα καλέσουμε την συνάρτηση ***newton_ raphson*** με αρχικό σημείο το x_0= **0** και παίρνουμε τα εξής αποτελέσματα:

```
Root = 0.00000
Iterations = 0
```

Σχήμα 16: Αποτελέσματα κλήσης της τροποποιήμενης συνάρτησης ***newton_ raphson*** με αρχικό σημείο $x_0 = \mathbf{0}$.

2.3 Μέθοδος της τέμνουσας

Η μέθοδος της τέμνουσας απότελει μία παραλλαγή της μεθόδου **Newton-Raphson** και χρησιμοποιείται συνήθως όταν είναι δύσκολο ή αδύνατο να υπολογιστεί η πρώτη παράγωγος **f(x)** και αλλάζει την πρώτη παράγωγο της **f(x)** στην αναδρομική ακολουθία της μεθόδου **Newton-Raphson** με την παρακάτω παράσταση :

$$\frac{f(x+h) \text{ - } f(x)}{(x+h)-x} \quad , \mathrm{h} \to 0$$

Οπότε τελικά η αναδρομική ακολουθία της μεθόδου της τέμνουσας είναι η παρακάτω :

$$x_{n+1} = x_n - \frac{f(x_n) * (x_n - x_{n-1})}{f(x_n) \text{ - } f(x_{n-1})}$$

Οι προϋποθέσεις που εγγυώνται την σύγκλιση της μεθόδου της τέμνουσας είναι ίδιες με αυτές της μεθόδου **Newton-Raphson** με την διαφορά ότι απαιτούνται δύο αρχικά σημεία x_0 και x_1. Συνήθως τα δύο αρχικά αυτά σημεία είναι τα άκρα του διαστήματος **[a,b]** όπου εφαρμόζουμε το θεώρημα **Bolzano**. Για την ρίζα της **f(x)** στο διάστημα **[-1.5,-1.0]** καλούμε την συνάρτηση ***secant*** από το αρχείο ***secant.py*** με ορίσματα την συνάρτηση **f(x)**, και τα δύο αρχικά σημεία x_0= **-1.5** και x_1= **-1.0** καθώς πληρούνται όλες οι προϋποθέσεις της μεθόδου της τέμνουσας σε αυτό το διάστημα, το **default** όρισμα **eps** έχει την τιμή που χρειάζεται για ακρίβεια **5** δεκαδικών ψηφίων ενώ το **default** όρισμα **max_iterations** αντιστοιχεί στον μέγιστο αριθμό επαναλήψεων που θα γίνουν σε περίπτωση που έχουν δοθεί αρχικά σημεία x_0 και x_1 χωρίς να τηρούνται οι προϋποθέσεις που εγγυώνται σύγκλιση , με τιμή τον αριθμό **50**. Αν πληρούνται όλες οι προϋποθέσεις η συνάρτηση επιστρέφει την ρίζα της **f(x)** στο αντίστοιχο διάστημα **(a,b)** (εδώ στο **(-1.5,-1.0)**), αφού οι προαναφερόμενες προϋποθεσεις εγγυώνται την ύπαρξη **μοναδικής** ρίζας στο διάστημα αυτό, καθώς και τον αριθμό των επαναλήψεων που χρειάστηκαν για να βρέθει η ρίζα με σφάλμα λιγότερο από **eps**.

```
root, iterations_num = secant(g, -1.5, -1.0)
```

Σχήμα 17: Παράδειγμα κλήσης της συνάρτησης ***secant*** με αρχικά σημεία $x_0 =$ **-1.5** και $x_1 =$ **-1.0** .

Μετά την κλήση της συνάρτησης ***secant*** τα αποτελέσματα είναι τα εξής:

```
Root = -1.19762
Iterations = 10
```

Σχήμα 18: Αποτελέσματα κλήσης της συνάρτησης ***secant*** με αρχικά σημεία $x_0 =$ **-1.5** και $x_1 =$ **-1.0** .

Όπου παρατηρούμε ότι η ρίζα της **f(x)** στο διάστημα **[-1.5,-1.0]** με ακρίβεια 5 δεκαδικών ψηφίων είναι η **-1.19762** καθώς και ότι η μέθοδος της τέμνουσας χρειάστηκε **10** επαναλήψεις για να επιτύχει την επιθυμητή ακρίβεια. Στην συνέχεια για την επόμενη ρίζα της **f(x)** επιλέγουμε το διάστημα **[1.3,1.9]** και καλούμε την συνάρτηση ***secant*** με ορίσματα την συνάρτηση **f(x)**, και τα δύο αρχικά σημεία x_0= **1.3** και x_1= **1.9**.

```
root, iterations_num = secant(g, 1.3, 1.9)
```

Σχήμα 19: Παράδειγμα κλήσης της συνάρτησης ***secant*** με αρχικά σημεία $x_0 =$ **1.3** και $x_1 =$ **1.9** .

Μετά την κλήση τα αποτελέσματα είναι τα εξής:

```
Root = 1.53013
Iterations = 8
```

Σχήμα 20: Αποτελέσματα κλήσης της συνάρτησης ***secant*** με αρχικά σημεία $x_0 =$ **1.3** και $x_1 =$ **1.9** .

Όπου παρατηρούμε ότι η ρίζα της $\mathbf{f(x)}$ στο διάστημα **[1.3,1.9]** με ακρίβεια 5 δεκαδικών ψηφίων είναι η **1.53013** καθώς και ότι η μέθοδος της τέμνουσας χρειάστηκε **8** επαναλήψεις για να επιτύχει την επιθυμητή ακρίβεια. Τέλος, με την μέθοδο της τέμνουσας αντιμετωπίζουμε το ίδιο πρόβλημα που αντιμετωπίζουν και οι δύο προαναφερόμενες μέθοδοι για την ρίζα $\mathbf{x = 0}$. Αν προσπαθήσουμε να καλέσουμε την συνάρτηση ***secant*** με αρχικά σημεία κάποια σημεία στα οποία δεν τηρούνται όλες οι προϋποθέσεις τα αποτελέσματα είναι μη προβλέψιμα και είτε μπορεί να έχουμε σύγκλιση σε λάθος αποτέλεσμα είτε σύγκλιση χωρίς την επιθυμητή ακρίβεια.

x_0	x_1	Αποτέλεσμα κλήσης συνάρτησης ***secant***	Αριθμός Επαναλήψεων
-0.25	0.25	-0.00002	44
-0.5	0.5	0.00008	46

Σχήμα 21: Αποτέλεσματα κλήσεων της συνάρτησης ***secant*** χωρίς να τηρούνται όλες οι προϋποθέσεις

Όπως παρατηρούμε και στο σχήμα 20, καμία κλήση δεν βρίσκει την ρίζα της $\mathbf{f(x)}$ με την επιθυμητή ακρίβεια. Η συνάρτηση ***secant*** που έχει υλοποιηθεί στο αρχείο ***secant.py*** έχει τροποποιηθεί έτσι ώστε να ελέγχει αν κάποιο από τα αρχικά σημεία x_0,x_1 αποτελεί ρίζα της $\mathbf{f(x)}$. Έτσι, αν τώρα καλέσουμε την συνάρτηση ***secant*** με αρχικά σημεία τα $x_0 = \mathbf{0}$ και $x_1 = \mathbf{1.5}$ παίρνουμε τα εξής αποτελέσματα:

```
Root = 0.00000
Iterations = 0
```

Σχήμα 22: Αποτελέσματα κλήσης της τροποποιήμενης συνάρτησης ***secant*** με αρχικά σημεία $x_0 = \mathbf{0}$ και $x_1 = \mathbf{1.5}$.

Όπου το $x_0 = \mathbf{0}$ αποτελεί ρίζα της $\mathbf{f(x)}$. Τέλος, παρόμοια με την μέθοδο της διχοτόμησης αν και τα δύο αρχικά σημεία x_0,x_1 αποτελούν ρίζες επιστρέφεται η τιμή του x_0.

2.4 Υπολογισμός σύγκλισης της μεθόδου Newton-Raphson για κάθε ρίζα

Για την μέθοδο **Newton-Raphson** ζητείται επίσης να δειχθεί για ποιες ρίζες συγκλίνει τετραγωνικά και για ποιες όχι. Αρχικά αναφέρουμε τις τρεις ρίζες της συνάρτησης **f(x)** με ακρίβεια 5ου ψηφίου. Στην συνέχεια αναφέρουμε το θεώρημα

x_0 = **-1.19762**
x_1 = **1.53013**
x_2 = **0.00000**

Σχήμα 23: Ρίζες της συνάρτησης f(x) στο διάστημα [-2,2]

που θα χρησιμοποιηθεί για τον υπολογισμό της τάξης σύγκλισης της κάθε ρίζας.
Θεώρημα : Έστω ότι $x_n \neq x^*$ για κάθε φυσικό αριθμό n, όπου x^* σταθερό σημείο της συνάρτησης **φ(x)** που χρησιμοποιείται στην μέθοδο σταθερού σημείου, και έστω ότι ισχύει :

$$\lim_{n\to\infty} \frac{x_{n+1} - x^*}{(x_n - x^*)^p} = a \neq 0,$$

τότε η τάξη σύγκλισης της ακολουθίας x_n είναι ακριβώς **p**.
Πιο συγκεκριμένα για την μέθοδο **Newton-Raphson** χρησιμοποιώντας το θεώρημα **Taylor** με κέντρο το σημείο x^*, όπου x^* ρίζα της **f(x)**, αποδεικνύεται ότι για την αναδρομική ακολουθία της μεθόδου **Newton-Raphson** το προηγούμενο όριο ισούται για $p = 2$ με :

$$\frac{f''(x^*)}{2f'(x^*)} \neq 0,$$

εφόσον $f'(x^*) \neq 0$.
Με αυτά που αναφέρθηκαν πλέον μπορούμε να προχωρήσουμε στον υπολογισμό της σύγκλισης της μεθόδου **Newton-Raphson** για κάθε μία από τις τρεις ρίζες.

Υπολογίζουμε αρχικά την πρώτη και την δεύτερη παράγωγο της **f(x)** στο διάστημα [**-2.2**]:

$$f'(x) = 3e^{sin^3 x} * sin^2 x * cosx + 6x^5 - 8x^3 - 3x^2$$

$$f''(x) = 9e^{sin^3 x} * sin^4 x * cos^2 x + 3e^{sin^3 x} * sin(2x) * cosx - 3e^{sin^3 x} * sin^3 x + 30x^4 - 24x^2 - 6x$$

- x_0 **= -1.19762**

  Ισχύει

  $$f'(-1.19762) = 35.62888 \neq 0,$$

  επομένως μπορούμε να χρησιμοποιήσουμε τον προηγούμενο τύπο για την σύγκλιση της μεθόδου **Newton-Raphson** για την ρίζα x_0. Οπότε έχουμε

  $$\lim_{n\to\infty} \frac{x_{n+1} - x_0}{(x_n - x_0)^2} = \frac{f''(x_0)}{2f'(x_0)} = \frac{35.62888}{-2 * 4.92044} = -3.62049 \neq 0$$

  επομένως η μέθοδος **Newton-Raphson** συγκλίνει τετραγωνικά για την ρίζα x_0 **= -1.19762**.

- x_1 **= 1.53013**

  Ισχύει

  $$f'(1.53013) = 14.97241 \neq 0,$$

  επομένως μπορούμε να χρησιμοποιήσουμε τον προηγούμενο τύπο για την σύγκλιση της μεθόδου **Newton-Raphson** για την ρίζα x_1. Οπότε έχουμε

  $$\lim_{n\to\infty} \frac{x_{n+1} - x_1}{(x_n - x_1)^2} = \frac{f''(x_1)}{2f'(x_1)} = \frac{91.03090}{2 * 14.97241} = 3.03995 \neq 0$$

  επομένως η μέθοδος **Newton-Raphson** συγκλίνει τετραγωνικά για την ρίζα $x_1 = \mathbf{1.53013}$.

- $x_2 = \mathbf{0.00000}$

  Ισχύει

  $$f'(0.00000) = 0.00000 = 0,$$

  επομένως δεν μπορούμε να χρησιμοποιήσουμε τον προηγούμενο τύπο για την σύγκλιση της μεθόδου **Newton-Raphson** για την ρίζα x_2. Αν υπολογίσουμε και την δεύτερη και τρίτη παράγωγο της $\mathbf{f(x)}$ παρατηρούμε ότι κι αυτές έχουν ρίζα το $x_2 = \mathbf{0.00000}$, επομένως η ρίζα x_2 έχει πολλαπλότητα $m = \mathbf{4}$. Χρησιμοποιώντας το ακόλουθο θεώρημα

  Θεώρημα : Έστω μία συνάρτηση $\mathbf{f(x)}$ $(m+1)$ φορές συνεχώς παραγωγίσιμη στο διάστημα **[a,b]** κι έστω ότι η συνάρτηση έχει ως ρίζα το σημείο $x = \mathbf{r}$ με πολλαπλότητα m, τότε η μέθοδος **Newton-Raphson** συγκλίνει τοπικά στο σημείο $x = \mathbf{r}$ και το σφάλμα e_i στην **i**-οστή επανάληψη ικανοποιεί το παρακάτω

  $$\lim_{n\to\infty} \frac{e_{i+1}}{x_i} = S,$$

  όπου

  $$S = \frac{m-1}{m}.$$

  Χρησιμοποιώντας το παραπάνω θεώρημα έχουμε $S = \frac{3}{4}$ οπότε η μέθοδος **Newton-Raphson** συγκλίνει γραμμικά με $e_{i+1} \approx \frac{3}{4} * e_i$. Αν η πολλαπλότητα μιας ρίζας $\mathbf{r}$ είναι γνωστή από πριν μπορούμε να βελτιώσουμε την απόδοση της κλασσικής μεθόδου **Newton-Raphson** με μία μικρή τροποποίηση :

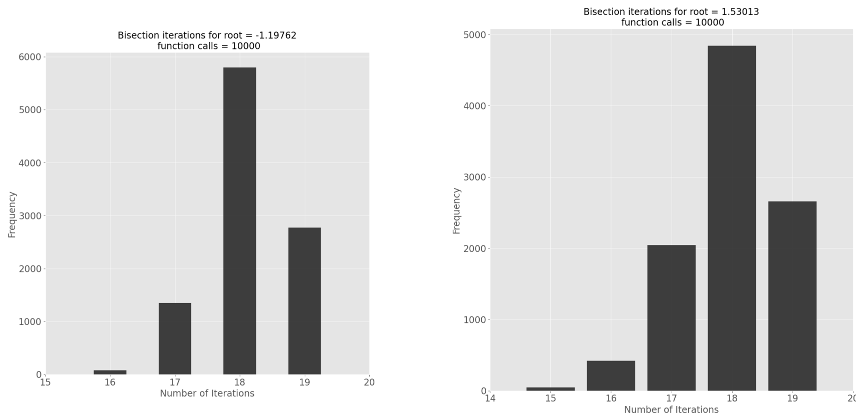
  $$x_{n+1} = x_n - \frac{m * f(x_n)}{f'(x_n)}$$

  όπου αποδεικνύεται ότι συγκλίνει τετραγωνικά στην ρίζα $\mathbf{r}$. Μία τέτοια συνάρτηση έχει υλοποιηθεί στο αρχείο ***mult_newton_raphson.py*** με όνομα ***mult_newton_raphson***, όπου έχει προστεθεί ένα ακόμα όρισμα **m** που δηλώνει την πολλαπλότητα της ρίζας που θέλουμε να υπολογιστεί από την συνάρτηση.

2.5 Σύγκριση επαναλήψεων για κάθε ρίζα

Σε αυτή την παράγραφο γίνεται μία σύγκριση των επαναλήψεων που χρειάστηκε κάθε μέθοδος για την εύρεση των τριών ριζών. Για να είναι τα αποτελέσματα της σύγκρισης αυτής πιο αντιπροσωπευτικά χρησιμοποιήθηκε δείγμα 10000 κλήσεων των συναρτήσεων που υλοποιούν την κάθε μέθοδο σε κατάλληλα διαστήματα και αρχικά σημεία και όχι τα μεμονωμένα ορίσματα που χρησιμοποιηθήκαν στις παραγράφους 2.1, 2.2 και 2.3. Εφόσον για την ρίζα $x= \mathbf{0}$ δεν μπορούν να βρεθούν κατάλληλα διαστήματα και αρχικά σημεία έτσι ώστε να τηρούνται οι προϋποθέσεις για κάθε μέθοδο, η ρίζα αυτή δεν συμπεριλαμβάνεται σε αυτή την ανάλυση. Εξάλλου με τον τρόπο που έχουν υλοποιηθεί οι συναρτήσεις στα αντίστοιχα αρχεία για όλες τις μεθόδους ο αριθμός των επαναλήψεων είναι 0 για την ρίζα $\mathbf{x} = \mathbf{0}$.

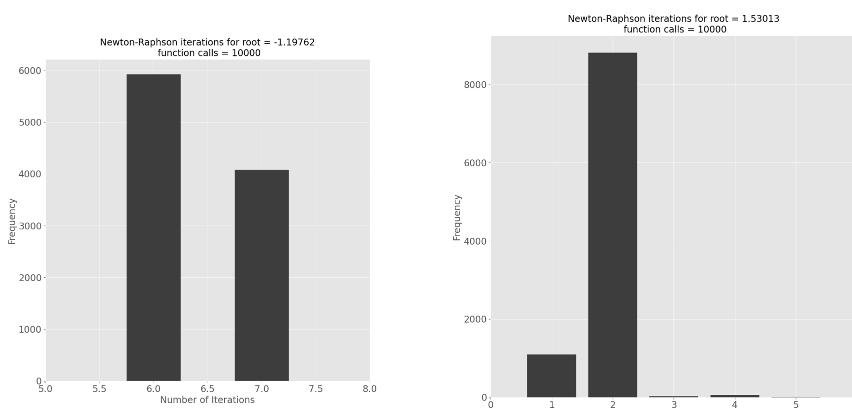
2.5.1 Μέθοδος της διχοτόμησης



Σχήμα 24: Ιστογράμματα συχνοτήτων του αριθμού των επαναλήψεων για την μέθοδο της διχοτόμησης.

Όπως παρατηρούμε από τα ιστογράμματα στο *Σχήμα 24* η μέθοδος της διχοτόμησης αποδίδει σε αριθμούς επαναλήψεων γύρω από το 18.

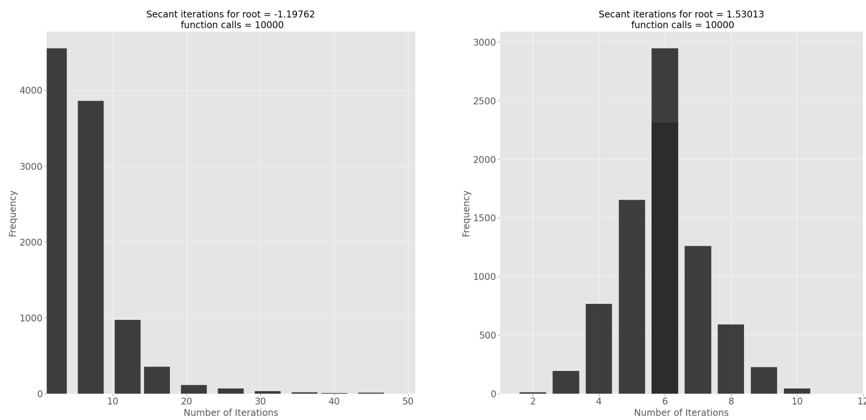
2.5.2 Μέθοδος Newton-Raphson



Σχήμα 25: Ιστογράμματα συχνοτήτων του αριθμού των επαναλήψεων για την μέθοδο Newton-Raphson.

Όπως παρατηρούμε από τα ιστογράμματα στο *Σχήμα 25* η μέθοδος Newton-Raphson αποδίδει κυρίως σε 2,6 και 7 αριθμούς επαναλήψεων.

2.5.3 Μέθοδος της τέμνουσας



Σχήμα 26: Ιστογράμματα συχνοτήτων του αριθμού των επαναλήψεων για την μέθοδο της τέμνουσας.

Όπως παρατηρούμε από τα ιστογράμματα στο *Σχήμα 26* η μέθοδος της τέμνουσας αποδίδει κυρίως σε αριθμούς επαναλήψεων κάτω από 10, με τις περισσότερες απάυτές να είναι γύρω από το 6, αλλά σε κάποιες περιπτώσεις (περίπου **20%** του δείγματος) ξεπερνάει τις 10.

Συμπερασματικά, έχουμε σύμφωνα με το δείγμα ότι έφοσον πληρούνται οι αντίστοιχες προϋποθέσεις για όλες τις μεθόδους η αποδοτικότερη και πιο σταθερή στην απόδοση της από τις μεθόδους ως προς τον αριθμό των επαναλήψεων είναι η μέθοδος **Newton-Raphson**. Στην συνέχεια, λιγότερο αποδοτικότερη είναι η μέθοδος της **τέμνουσας** ενώ τέλος την χειρότερη απόδοση έχει η μέθοδος της **διχοτόμησης**. Εξάλλου, η τάξη σύγκλισης της μεθόδου **Newton-Raphson** είναι $\mathbf{p = 2}$, της μεθόδου της **τέμνουσας** $\mathbf{p \approx 1.62}$ ενώ της μεθόδου της **διχοτόμησης** $\mathbf{p = 1}$, οπότε τα συμπεράσματα από τα ιστογράμματα αιτιολογούνται από τις τάξεις σύγκλισεις της κάθε μεθόδου.

3 Άσκηση 2