Αναφορά πρώτης εργαστηριακής άσκησης

HPY 419 - HPY 608 - Ανάπτυξη Εργαλείων CAD για Σχεδίαση Ολοκληρωμένων Κυκλωμάτων 2022- 2023

ΑΙΚΑΤΕΡΙΝΗ ΤΣΙΜΠΙΡΔΩΝΗ:2018030013

Εισαγωγή:

Ο σκοπός αυτής της εργασίας ήταν να υλοποιήσουμε με τη μέθοδο Newton- Raphson ένα πρόγραμμα το οποίο θα βρίσκει τις ρίζες σε μία πολυωνυμική εξίσωση.Αυτό το πρόγραμμα υπολογίζει τη ρίζα μιας πολυωνυμικής συνάρτησης χρησιμοποιώντας δύο διαφορετικές μεθόδους: την αριθμητική μέθοδο της εφαπτομένης και την αναλυτική μέθοδο για την εύρεση της παραγωγού. **Η εργασία ζητάει το πρόγραμμα να ασχολείται περισσότερο με πολυωνυμικές εξισώσεις πέμπτου βαθμού**.

Το πρόγραμμα λαμβάνει ως είσοδο τον βαθμό και τους συντελεστές της πολυωνυμικής συνάρτησης και επιστρέφει τη ρίζα της πολυωνυμικής συνάρτησης και τον αριθμό των αριθμητικών πράξεων που χρησιμοποιήθηκαν. Το πρόγραμμα υλοποιήθηκε σε γλώσσα C.

Σχεδιασμός και υλοποίηση:

Αυτή η υλοποίηση περιέχει αρκετές συναρτήσεις για να επιτευχθεί ο σκοπός, Αυτές είναι οι παρακάτω:

CalcPolF function:

Αυτή η συνάρτηση υπολογίζει την τιμή της πολυωνυμικής συνάρτησης σε ένα συγκεκριμένο σημείο. Τα ορίσματα εισόδου είναι η τιμή **x**, ενας πίνακας συντελεστών του πολυωνύμου **numOfPol** ,ο βαθμος του πολυωνύμου d και μία μεταβλητή mode . Η μεταβλητή mode βοηθάει στο να υπολογιστούν τα στατιστικά που μας ζητάει η άσκηση και που αφορούν τις προσθέσεις ,αφαιρέσεις, διαιρέσεις και τους πολλαπλασιασμούς που γίνονται. Δηλαδή όταν είναι ίση με 1 τότε θα αυξηθούν οι counters που μετράνε τα στατιστικά που αφορούν την μέθοδο αριθμητικής παραγωγού με την μέθοδο της εφαπτομένης, ενώ όταν είναι 0 θα αυξηθούν οι counters που αφορούν τα στατιστικά της αναλυτικής μεθόδου. Υπάρχει και μία περίπτωση πού το mode γίνεται ίσο με 2 διότι υπάρχει και μία συνάρτηση του λέγεται **findX0** και που καλεί την **CalcPolF** και για την οποία δεν χρειάζεται να μετρήσουμε τα στατιστικά εφόσον δεν επηρεάζουν κάπου το αποτέλεσμα διότι καλείται μόνο μία φορά στην main και υπολογίζει ένα αποτέλεσμα που χρησιμοποιείται και στις δύο συναρτήσεις.

Η συνάρτηση επιστρέφει την τιμή της συνάρτησης στο σημείο x, Επίσης να σημειωθεί ότι για τον υπολογισμό αυτόν χρησιμοποιήθηκε η μέθοδος **Horner**.

CalculateDer function:

Αυτή η συνάρτηση υπολογίζει την παράγωγο της πολυωνυμικης συναρτησης σε ένα συγκεκριμένο σημείο x , χρησιμοποιώντας τη μέθοδο της εφαπτομένης. Τα ορίσματα εισόδου είναι η τιμή της f(x) η τιμή της f(x+delta) και η τιμή delta που εμείς έχουμε διαλέξει. η συνάρτηση επιστρέφει την παράγωγο της συνάρτησης στο σημείο x η οποία έχει υπολογιστεί με την προσεγγιστική μέθοδο που μας δίνεται στην εκφώνηση για τη μέθοδο της εφαπτομένης.

findX0 function:
Αυτή η συνάρτηση βρίσκει την αρχική εικασία για τη ρίζα χρησιμοποιώντας τη μέθοδο της διχοτόμησης τα ορίσματα εισόδου είναι ο βαθμός του πολυωνύμου και ο πίνακας συντελεστών του πολυωνύμου. Δηλαδη η συνάντηση παράγει είκοσι τυχαίους αριθμούς στην περιοχή[-20,20] και υπολογίζει τις τιμές του πολυωνύμου σε αυτά τα σημεία. Αν υπάρχουν δύο σημεία με αντίθετα πρόσημα επιστρέφετε το μέσο σημείο μεταξύ τους ως αρχική εκτίμηση. Διαφορετικά η συνάρτηση επιστρέφει -10. Παρατήρησα ότι αν και έχουμε βάλει **rand()** η αρχική εκτίμηση του x0 σε μία συγκεκριμένη συνάρτηση θα είναι πάντα ίδια για ένα συγκεκριμένο περιβάλλον στο οποίο τρέχει ο κώδικας. Για μία συγκεκριμένη συνάρτηση το x0 δεν θα αλλάζει τιμή από την μία εκτέλεση προγράμματος στην επόμενη θα αλλάζει όμως από υπολογιστή σε υπολογιστή ή από περιβάλλον σε περιβάλλον εκτέλεσης κώδικα. Οπότε καλύπτεται το ζητούμενο Το x0 να είναι σταθερό για μία συγκεκριμένη συνάρτηση, προφανώς θα αλλάζει από συνάρτηση σε συνάρτηση. Αυτό πιθανότατα οφείλεται στο ότι το rand που χρησιμοποιήθηκε δεν είναι ακριβώς τυχαίο αλλά ψευδοτυχαίο .
NewtonRaphsonMethod function:
Αυτή η συνάρτηση υλοποιεί τη μέθοδο Newton-raphson για την εύρεση της λύσης της εξίσωσης, τα ορίσματα εισόδου είναι η τιμή της f(x) η παράγωγος της f(x), η τρέχουσα εικασία για τη ρίζα xk, Και το flag mode το οποίο εξηγήθηκε παραπάνω ποια είναι η χρησιμότητα του. Η συνάρτηση αυτή στην ουσία παίρνει μία εικασία αρχική x για τη ρίζα και επιστρέφει μία νέα εικασία.Αυτή η συνάρτηση θα καλεστεί αργότερα μέσα σε μία άλλη συνάρτηση επαναλαμβανόμενα ώστε να βρεθεί η πραγματική ρίζα.
CalcforNewRapsMethodForDelta function:
Αυτή η συνάρτηση υλοποιεί τη μέθοδο Newton raphson με την αριθμητική μέθοδο εύρεσης της παραγωγού χρησιμοποιώντας τη μέθοδο της εφαπτομένης για μία δεδομένη τιμή του Δ τα ορίσματα εισόδου είναι η τρέχουσα εικασία για τη ρίζα, ο βαθμός του πολυωνύμου ,ενας πίνακας συντελεστών του πολυωνύμου η τιμή του Δ που έχουμε επιλέξει και ο τρέχων αριθμός γύρου. Η συνάρτηση υπολογίζει την τιμή της f(x) , την τιμή της f(x+delta) καθώς και την παραγωγό της f(x) Χρησιμοποιώντας για τα δύο πρώτα τη συνάρτηση CalcPolF και για την παράγωγο την **CalculateDer**. Στη συνέχεια καλεί τη συνάρτηση **NewtonRaphsonMethod** για να ενημερώσει την εικασία για τη ρίζα. Η συνάρτηση επιστρέφει μία μεταβλητή struct που περιέχει την καινούργια εικασία για τη ρίζα την τιμή της f(x) για αυτή την ρίζα και τον τρέχοντα αριθμό των επαναλήψεων(γύρων).
CalculateRootWithDelta function:
Συνάρτηση βρίσκει τη ρίζα της εξίσωσης χρησιμοποιώντας τη μέθοδο Newton raphson με **delta=0,0001**. Τα ορίσματα εισόδου είναι η αρχική εικασία για τη ρίζα, ο βαθμός πολυωνύμου και ο πίνακας συντελεστών του πολυωνύμου. Η συνάρτηση καλεί τη συνάρτηση **CalcforNewRapsMethodForDelta** σε βρόχο μέχρι να βρεθεί η ρίζα. Η συνάρτηση εκτυπώνει την τιμή της ρίζας τον αριθμό των επαναλήψεων, το συνολικό αριθμό των πρόσθετων αφαιρέσεων το συνολικό αριθμό των πολλαπλασιασμων και τέλος το συνολικό αριθμό των διαιρέσεων.
CalcforNewRapsMethodForAn function:
Αυτή η συνάρτηση αφορά την αναλυτική εύρεση της παραγωγού του πολυωνύμου και αφορά επίσης τη μέθοδο Newton Raphson για την εύρεση της ρίζας. Τα ορίσματα εισόδου είναι η τρέχουσα εικασία για τη ρίζα ,ο βαθμός του πολυωνύμου, ο πίνακας συντελεστών του πολυωνύμου ,επίσης ο πίνακας συντελεστών της παραγωγού του πολυωνύμου και ο τρέχων αριθμός επαναλήψεων. Το πολυώνυμο αναπαριστάται ως ένας πίνακας συντελεστών που ονομάζεται **numOfPol** και η παράγωγος του πολυωνύμου ως Ένας πίνακας συντελεστών

που ονομάζεται **NumOfPolD**,ο βαθμός του πολυωνύμου δίνεται από το όρισμα Degree. Ο κώδικας χρησιμοποιεί μία συνάρτηση για τον υπολογισμό της τιμής του πολυωνύμου καθώς και της παραγωγού του πολυωνύμου σε ένα συγκεκριμένο σημείο και αυτή είναι η συνάρτηση **CalcPolF,** η οποία από μόνη της υπολογίζει την τιμή μιας συνάρτησης f(x) για ένα συγκεκριμένο x, Οπότε για να υπολογίσουμε τη πολυωνυμική συνάρτηση βάζουμε ως όρισμα τον βαθμό τις συναρτήσεις καθώς και τους συντελεστέςπου αποθηκεύτηκαν στον πίνακα numOfPol.Ενώ για να υπολογίσουμε την παραγωγό βάζουμε τον βαθμό της πολυωνυμική συνάρτηση - 1 όπως ξέρουμε από τα μαθηματικά και τους συντελεστές που αποθηκεύτηκαν στον **NumOfPolD**(αυτοί οι δύο πίνακες όπως θα δούμε αργότερα κατασκευάζονται μέσα στη main). Στη συνέχεια καλεί τη συνάρτηση **NewtonRaphsonMethod** για να ενημερώσει την εικασία για τη ρίζα. Η συνάρτηση επιστρέφει μία μεταβλητή struct που περιέχει την καινούργια εικασία για τη ρίζα την τιμή της f(x) για αυτή την ρίζα και τον τρέχοντα αριθμό των επαναλήψεων(γύρων).

CalculateRootWithAn function:

Η συνάρτηση βρίσκει τη ρίζα της εξίσωσης χρησιμοποιώντας τη μέθοδο Newton Paphson.Τα ορίσματα εισόδου είναι η αρχική ρίζα ,ο βαθμός πολυωνύμου και οι πίνακες που έχουμε αποθηκεύσει τους συντελεστές του πολυωνύμου και της παραγωγού του πολυωνύμου.Η συνάρτηση **CalcforNewRapsMethodForAn** καλείται σε βρόγχο μέχρι να βρεθεί η ρίζα και εκτυπώνει την τιμή της ρίζας καθώς και τα στατιστικά που μας ζητάει η άσκηση .

main():

Ελέγχει τη ροή του προγράμματος. Ζητάει από το χρήστη να του δώσει ως είσοδο το βαθμό και τους συντελεστές της πολυωνυμικής συνάρτησης .Αποθηκεύει το βαθμό του πολυωνύμου σε μία μεταβλητή που ονομάζεται **Degree** και τους συντελεστές σε μία μεταβλητή που ονομάζεται **NumOfPol**.Εκτυπώνει το πολυώνυμο κατασκευάζει τους συντελεστές της παραγωγού και τις αποθηκεύεις σε ένα πίνακα **NumOfPolD**.Επίσης εκτυπώνει και την παραγωγό το οποίο δεν το ζητάει ως έξοδο απλά ήταν μία μορφή επαλήθευσης .Ύστερα καλεί τη συνάρτηση **findX0** για να βρει μία αρχική εκτίμηση του x και τέλος θα χρησιμοποιήσει αυτή την αρχική εκτίμηση ώστε να υπολογίσει τη ρίζα είτε βρίσκοντας την παραγωγό με την αριθμητική μέθοδος εφαπτομένης είτε αναλυτικά. Για να υπολογίσει την ρίζα με την αριθμητική μέθοδο καλεί τη συνάρτηση **CalculateRootWithDelta**. Ενω για να υπολογίζει τη ρίζα αναλυτική παραγωγό καλεί **CalculateRootWithAn**. Οι Συναρτήσεις αυτές που έχουν εξηγηθεί διεξοδικά παραπάνω.

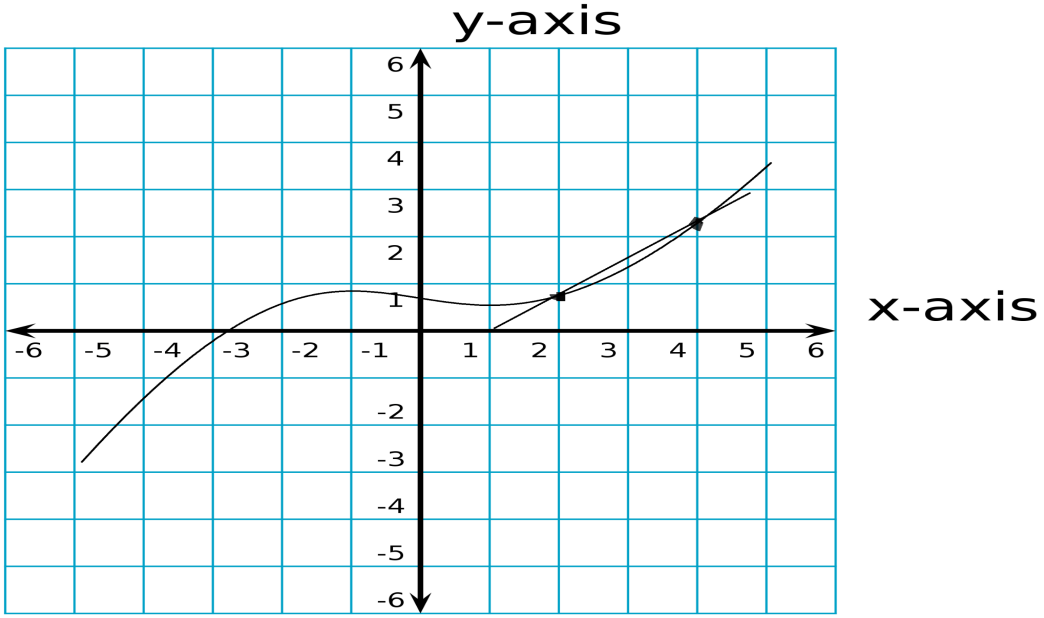
Δοκιμές και ανάλυση

Το πρόγραμμα δοκιμάστηκε με διάφορες εισόδους, κυρίως σε πολυώνυμα πέμπτου βαθμού με διαφορετικούς συντελεστές.

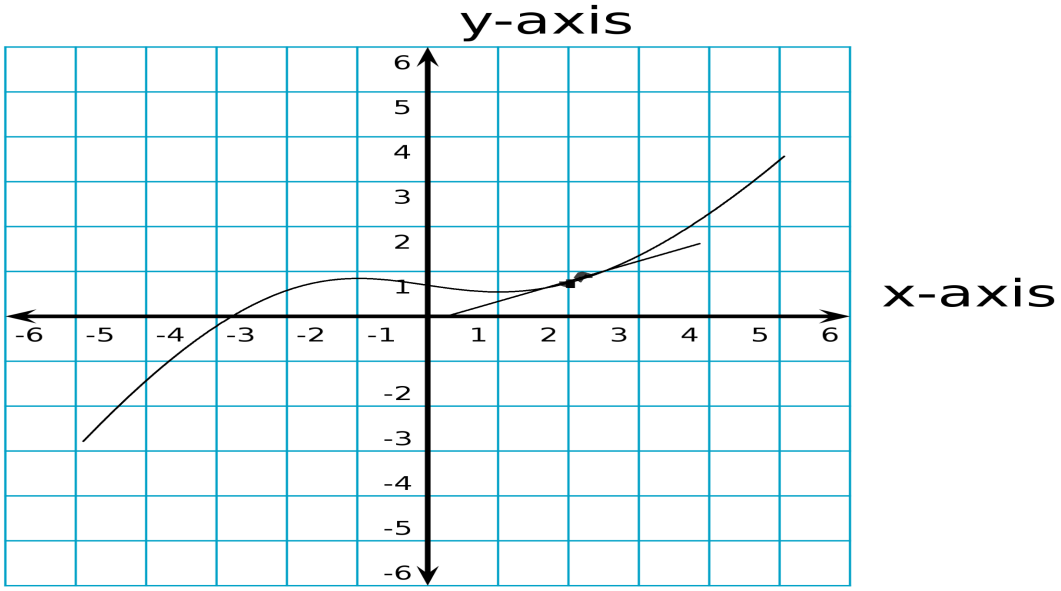
Το πρόγραμμα χρησιμοποιεί μετρητές για την παρακολούθηση του αριθμού των αριθμητικών πράξεων που χρησιμοποιούνται σε κάθε μέθοδο. Οι μετρητές είναι οι AddSubCounterD και MultDCounter για τη λειτουργία δέλτα και οι AddSubCounterA και MultACounter για την αναλυτική λειτουργία. Το πρόγραμμα χρησιμοποιεί επίσης μετρητές για την παρακολούθηση του αριθμού των διαιρέσεων, οι οποίοι είναι οι divDCounter για τη λειτουργία delta και divACounter για την αναλυτική λειτουργία.

Η αποδοτικότητα του προγράμματος αξιολογήθηκε ως προς τον αριθμό των αριθμητικών πράξεων που χρησιμοποιούνται από κάθε μέθοδο. Τα αποτελέσματα δείχνουν ότι η αναλυτική μέθοδος απαιτεί λιγότερες αριθμητικές πράξεις από τη μέθοδο με δέλτα. Αυτό οφείλεται στο γεγονός ότι η αναλυτική μέθοδος χρησιμοποιεί την παράγωγο του

πολυωνύμου για τον υπολογισμό της ρίζας, ενώ η μέθοδος με δέλτα απαιτεί τον υπολογισμό του πολυωνύμου σε πολλά σημεία για την εκτίμηση της παραγώγου.
Η μέθοδος της εφαπτομένης ή αλλιώς μέθοδος δελτα (xn+1 = xn – f(xn) / {[(f(xn+δ) – f(xn)] / δ } μπορεί να χρειαστεί περισσότερες επαναλήψεις ώστε να βρεθεί η ρίζα και αυτό που Παρατηρήθηκε είναι ότι όσο μεγαλώνει το τιμή του δ τόσες περισσότερες επαναλήψεις έχουμε και αυτό είναι λογικό διότι το δ στην ουσία είναι πόσο απέχει η εκτιμώμενη ρίζα από την επόμενη όσο πιο κοντά είναι αυτές οι δύο ρίζες τόσο μεγαλύτερη ακρίβεια έχουν.
Ένα παράδειγμα παρακάτω από παρατηρούμε ότι η απόσταση των δύο διαδοχικών Χ είναι δύο οπότε το δ=2 εδώ βλέπουμε ότι τραβώντας την εφαπτομένη σε αυτά τα δύο σημεία δηλαδή την παραγωγό έχει αρκετή απόκλιση από την πραγματική συνάρτηση και αυτό σίγουρα θα μας στοιχίσει σε επαναλήψεις και βέβαια και σε αριθμητικές πράξεις αφού θα έχουμε περισσότερες επαναλήψεις.



Ενώ αν το δ είναι πολύ μικρό έχουμε το παρακάτω σχήμα:

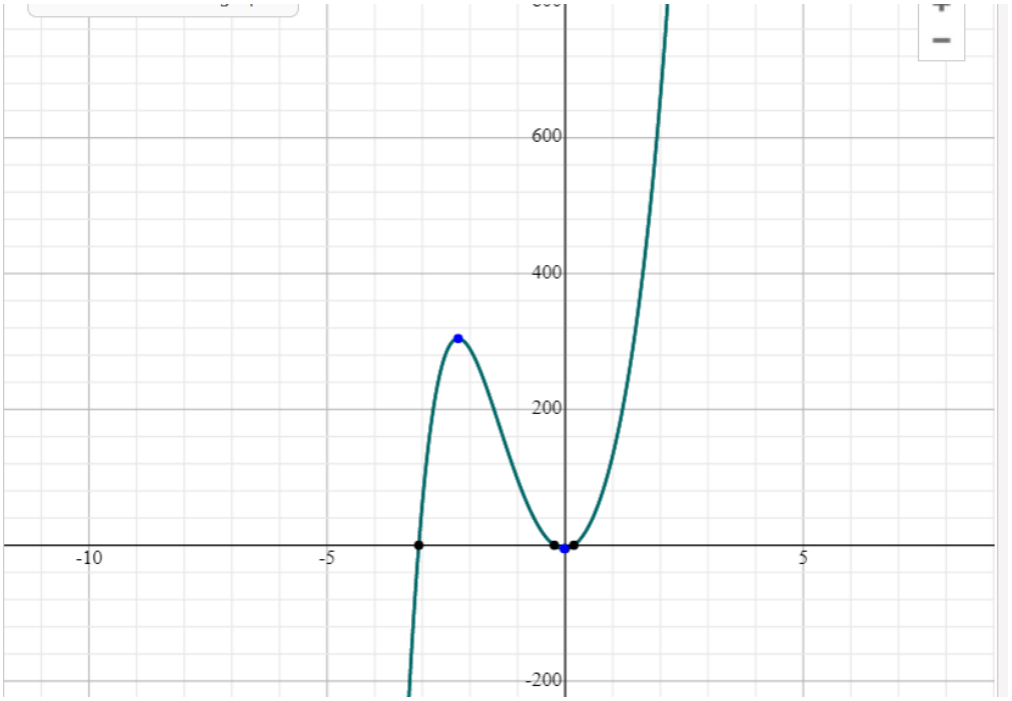


Στο οποίο φαίνεται ξεκάθαρα ότι η επόμενη εκτίμηση για x που θα κάνουμε βρίσκεται πιο κοντά στη ρίζα από την πάνω διότι η εφαπτομένη των δύο σημείων δηλαδή παραγωγός

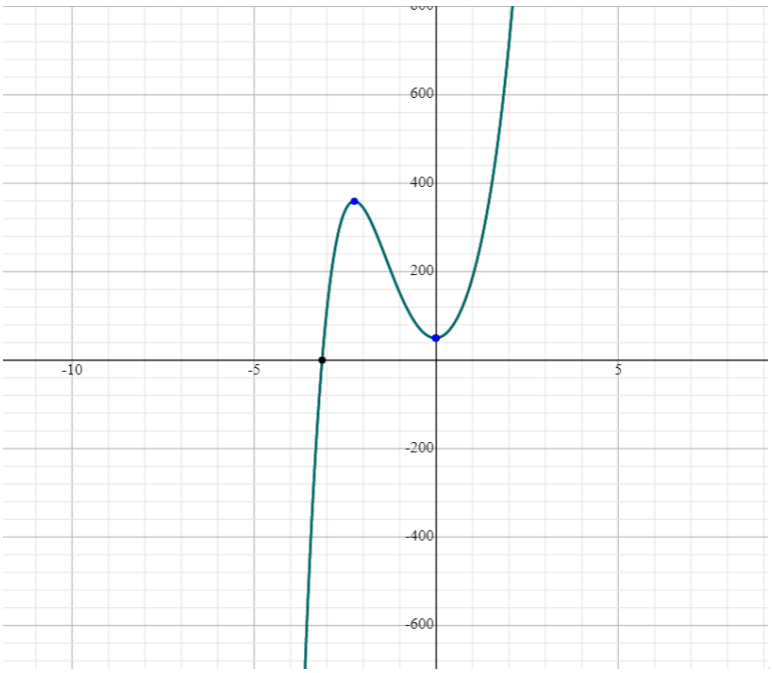
σχεδόν εφάπτεται με την συνάρτηση.Και αυτό μας γλιτώνει από επαναλήψεις και κατά συνέπεια από περισσότερες αριθμητικές πράξεις.
Στο δικό μου πρόγραμμα δοκίμασα αρκετές τιμές του Δ για διάφορα πολυώνυμα για την ακρίβεια ξεκίνησα από το 1 μετά πήγα 0.1 0.01κτλ και είδα ότι τις λιγότερες επαναλήψεις έχουμε για 0.0001 δοκιμάστηκε και το 0.00001 αλλά παρατηρήθηκε ότι δεν αλλάζει τίποτα οπότε προτιμήθηκε το πρώτο.
Σημειώσω εδώ ότι δεν συμπεριλήφθηκαν στη μέτρηση των αριθμητικών πράξεων οι πράξεις που συμβαίνουν μέσα στη συνάρτηση findX0 δεν το θεωρήθηκε σκόπιμο επειδή θα είναι και για τις δύο μεθόδους ίδιος αριθμός οπότε στην ουσία θα αύξανε τις αριθμητικές πράξεις και των δύο μεθόδων αλλά δεν θα έπαιζε ρόλο στη σύγκριση τους.

Δοκιμές:

Αρχικά έψαξα να βρω μία συνάρτηση η οποία να έχει παραπάνω από μία ριζες πχ. 3x^5+10x^3+120x^2+5x-5=0 , η οποία έχει 3 ρίζες



και μετά από απεικόνιση μετατοπίστηκε στον άξονα y προσθέτοντας της τον αριθμό 45 .
Έτσι τώρα έχουμε τη συνάρτηση 3x^5+10x^3+120x^2+5x+50=0 η οποία έχει μία ρίζα



Δόθηκε ως είσοδο στο πρόγραμμα μας και βγάζει το ακόλουθο αποτέλεσμα.

```
derivative with arithmetic method delta:
 round: 1
 x0 = -11.50000

 round: 2
 xk = -9.22008

 round: 3
 xk = -7.40376

 round: 4
 xk = -5.97446

 round: 5
 xk = -4.87463

 round: 6
 xk = -4.05962

 round: 7
 xk = -3.51306

 round: 8
 xk = -3.22380

 round: 9
 xk = -3.13885

 round: 10
 xk = -3.13214

 round: 11
 xk = -3.13211

 round: 12
 xk = -3.13211

Root is: -3.13211
Total rounds: 12
Total add/sub: 132
Total multiplications: 110
Total division: 22
```

```
Total rounds: 12
Total add/sub: 132
Total multiplications: 110
Total division: 22
derivative with analytic method:
 round: 1
 x0 = -11.50000

 round: 2
 xk = -9.21306

 round: 3
 xk = -7.39809

 round: 4
 xk = -5.97073

 round: 5
 xk = -4.86969

 round: 6
 xk = -4.05559

 round: 7
 xk = -3.50994

 round: 8
 xk = -3.22259

 round: 9
 xk = -3.13883

 round: 10
 xk = -3.13215

 round: 11
 xk = -3.13211

 round: 12
 xk = -3.13211

Root is: -3.13211
Total rounds: 12
Total add/sub: 110
Total multiplications: 99
Total division: 11
Process returned 19 (0x13)   execution time : 40.707 s
Press any key to continue.
```

Βλέπουμε εδώ ότι βρίσκει τη ρίζα και από όσο το έλεγξα είναι σωστή. Επίσης ο αριθμός των round για να φτάσει στη ρίζα είναι ακριβώς ο ίδιος που σημαίνει ότι το δέλτα που επιλέχθηκε είναι αρκετά καλό . Τέλος όπως έχει αναφερθεί και πάνω ότι όντως η πρώτη μέθοδος έχει περισσότερες πράξεις από τη δεύτερη.

Προφανώς και δοκίμασα τη συνάρτηση που αναφέρεται στην εκφώνηση και μου έβγαζε το παρακάτω αποτέλεσμα:

derivative with arithmetic method delta:

```
Root is: -1.06348
Total rounds: 15
Total add/sub: 168
Total multiplications: 140
Total division: 28
derivative with analytic method:
```

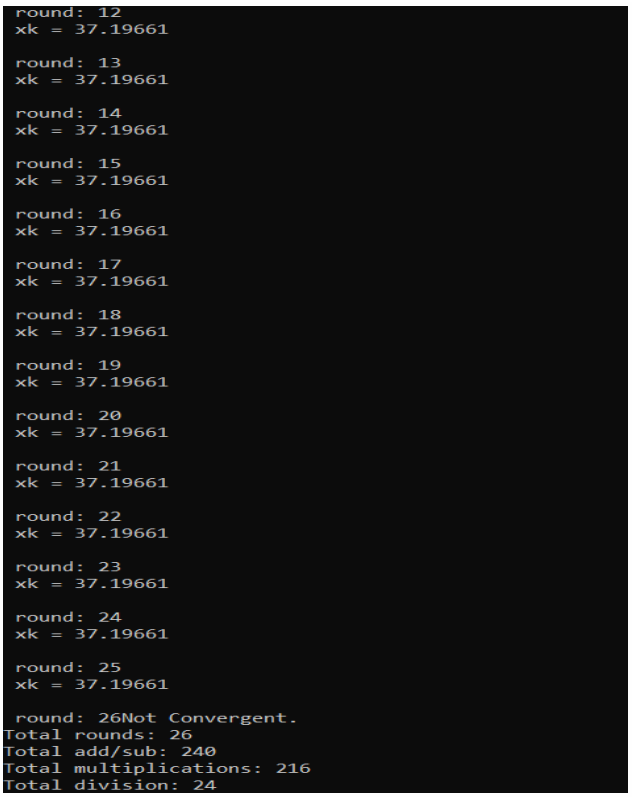
derivative with analytic method:

```
Root is: -1.06348
Total rounds: 15
Total add/sub: 140
Total multiplications: 126
Total division: 14
```

Η ρίζα είναι αυτη που πρέπει και ότι το πρόγραμμα δουλεύει σωστά τουλάχιστον για τα πολυώνυμα που έχουν μία ρίζα.
Τέλος χρησιμοποιήθηκε ως είσοδος η συνάρτηση που δόθηκε για δοκιμή
f(x)=3x**5 -124x**4+439x**3+1234x**2+10439x-931678 και τα αποτελέσματα είναι τα παρακάτω
derivative with arithmetic method delta:

```
xk = 37.19661

round: 14
xk = 37.19661

round: 15
xk = 37.19661

round: 16
xk = 37.19661

round: 17
xk = 37.19661

round: 18
xk = 37.19661

round: 19
xk = 37.19661

round: 20
xk = 37.19661

round: 21
xk = 37.19661

round: 22
xk = 37.19661

round: 23
xk = 37.19661

round: 24
xk = 37.19661

round: 25
xk = 37.19661

round: 26
Not Convergent.
Total rounds: 26
Total add/sub: 288
Total multiplications: 240
Total division: 48
derivative with analytic method:
```

derivative with analytic method:



Η συνάρτηση δεν κατάφερε να συγκλίνει σε μία ρίζα μετά από το πέρας των 25 γύρων που έχει οριστεί ως άνω όριο επαναλήψεων και μάλλον με αυτή τη μέθοδο δεν θα καταφέρει να συγκλίνει εύκολα διότι όπως φαίνεται παραπάνω για αρκετούς γύρους παραμένει σε ένα σημείο, όταν απεικονίστηκε η συνάρτηση παρατηρήθηκε ότι αυτό το σημείο είναι αρκετά κοντά στην πραγματική ρίζα που έχει όμως εδώ δεν καταφέρνει f(x)να γίνει μηδέν.

Είναι από αυτές οι περιπτώσεις πού μάλλον η παραπάνω μέθοδος δεν μπορεί να δώσει λύση.