ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

[HRY 419]-Ανάπτυξη Εργαλείων CAD για Σχεδίαση Ολοκληρωμένων Κυκλωμάτων

Αναφορά 1ου εργαστηρίου

Ιωάννης Περίδης
A.M. 2018030069

9 Μαρτίου 2021

1 Εισαγωγή:

Σκοπός της εργαστηριακής αυτής άσκησης, ήταν η υλοποίηση και η μελέτη θεμάτων σύγκλισης , αριθμού επαναλήψεων, αλλά και αριθμητικής προσέγγισης (μέθοδος τέμνουσας) στην πρώτη παράγωγο σε σχέση με αναλυτική προσέγγιση. Συγκριμένα, μας ζητήθηκε η εύρεση της μοναδικής ρίζας ενός πολυωνύμου έως και 5ου βαθμού, με δύο τρόπους, με την μέθοδο Newton-Raphson αναλυτικά και ύστερα με την προσεγγιστική μέθοδο Secant .Η εργασία υλοποιήθηκε σε γλώσσα C η οποία είναι procedural και structed και ταυτόχρονα ο κώδικας έγινε όσο το δυνατό περισσότερο strongly typed.

2 Υλοποίηση:

Ξεκινώντας, το πρόγραμμα ζητάει από τον χρήστη να δώσει σαν είσοδο τον βαθμό του πολυωνύμου, με μέγιστο επιτρεπτό βαθμό το 5. Ύστερα, ανάλογα τον βαθμό ο χρήστης πρέπει να εισάγει το αντίστοιχο πλήθος συντελεστών του πολυωνύμου (συντελεστές όσοι ο βαθμός +1 για την σταθερά).Στην συνέχεια το πρόγραμμα εκτυπώνει το ολοκληρωμένο πολυώνυμο στην οθόνη και προχωράει στον υπολογισμό των 2 μεθόδων.

3 Μέθοδος Newton-Raphson:

Η μέθοδος Newton-Raphson, είναι μια μέθοδος διαδοχικών προσεγγίσεων για την προσεγγιστική εύρεση των ριζών μιας πραγματικής συνάρτησης. Η μέθοδος όταν συγκλίνει, συγκλίνει ιδιαίτερα γρήγορα και πιο συγκεκριμένα τετραγωνικά. Παρόλα αυτά, δεν εγγυάται πάντοτε την σύγκλιση. Σημαντικός παράγοντας την ύπαρξη σύγκλισης , είναι η επαναληπτική διαδικασία να ξεκινάει με μια εκτίμηση της πρώτης ρίζας x0 η οποία να είναι «κοντά» στην πραγματική ρίζα. Αυτό διότι, αν είναι πολύ μακριά από την πραγματική ίσως η σύγκληση να αργήσει πολύ ή ακόμα και να μην συγκλίνει καθόλου μέσα σε ένα πλαίσιο συγκεκριμένων επαναλήψεων. Το ερώτημα το πόσο «κοντά» πρέπει να είναι, θα εξεταστεί παρακάτω με το πρόγραμμα με δοκιμές.

Θα ασχοληθούμε με συναρτήσεις (κυρίως περιττού βαθμού πολυώνυμα) οι οποίες θα έχουν μονάχα μια ρίζα στον άξονα χ.

Ο τύπος για την εύρεση της πρώτης προσέγγισης x1, με ένα δεδομένο τυχαίο x0 ,τιμή f(x0) και παράγωγο f ‘(x0) σημείο δίνεται:

$$\boldsymbol{x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}} \ (\mathbf{1})$$

Επομένως, η γενική αναδρομική σχέση της μεθόδου είναι:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad (2)$$

Η υλοποίηση της μεθόδου ήταν αρκετά απλή. Δημιουργήθηκαν αρχικά, δύο συναρτήσεις fp και fpDerivative οι οποίες παίρνουν σαν όρισμα το πολυώνυμο και μια συγκεκριμένη θέση του και υπολογίζουν την τιμή του και την παράγωγο της τιμής του αναλυτικά σε αυτό το σημείο.

Στην συνέχεια, μέσα σε έναν βρόγχο έως και 25 επαναλήψεων ,υλοποιήθηκε με την βοήθεια των συναρτήσεων ο παραπάνω τύπος (1) με αρχική τιμή x0=1. Για να δημιουργηθεί η γενική αναδρομή , απλά στο τέλος της κάθε επανάληψης δίνεται η τιμή του x1 στο x0. Ο βρόγχος τερματίζει με δύο τρόπους. Τερματίζει επιτυχώς, δηλαδή με σύγκλιση σε μια ρίζα xn αν εντός των 25 επαναλήψεων αυτών γίνει αληθής η συνθήκη τερματισμού : $|f(x_n)| < e = 10^{-3}$ η οποία μας δίνει μια επαρκής προσέγγιση αποδεκτού σφάλματος 0,1%. Αλλιώς αν ξεπεραστούν οι 25 κύκλοι και δεν έχει γίνει αληθής η συνθήκη , τότε το πολυώνυμο δεν συγκλίνει σε κάποια ρίζα και η μέθοδος τερματίζει ανεπιτυχώς.

Το πρόγραμμα δίνει σαν έξοδο την πρώτη εκτίμηση χ0, την ρίζα του πολυωνύμου (αν συγκλίνει, αλλιώς δίνει μήνυμα ανεπιτυχής), το πλήθος των επαναλήψεων που χρειάστηκαν και το xk της κάθε επανάληψης και το κόστος της μεθόδου. Συγκεκριμένα υπολογίζεται το πλήθος των πολλαπλασιασμών, των διαιρέσεων και των προσθαφαιρέσεων που έγιναν για τον υπολογισμό (αυτό συμπεριλαμβάνει το κόστος για την δημιουργία του τύπου(1), όπως και των συναρτήσεων που κλήθηκαν ΚΑΙ τον έλεγχο της συνθήκης σε κάθε βρόγχο , δηλαδή ακόμη μια κλήση της συνάρτησης fp).

Παρακάτω φαίνεται μια φωτογραφία ενός αποτελέσματος για το πολυώνυμο :

$\boldsymbol{3\chi^5 + 4\chi^4 - 2\chi^3 + \chi - 10,}$ το οποίο έχει πραγματική ρίζα: $\boldsymbol{\chi = 1{,}12273}$ ...

```
Please enter the grade of your polynomial (max grade=5):
5
Please enter the 6 parameters of your polynomial.
If a parameter does not exist it should be enter as 0 and not be skipped:
3 4 -2 0 1 -10
The polynomial you entered is: (3)*x^5 + (4)*x^4 + (-2)*x^3 + (0)*x^2 + (1)*x + (-10)

*******Newton-Raphson*******:
The first root estimation xo=1.000000
For repetition 0 :xk=1.153846
For repetition 1 :xk=1.124257
For repetition 2 :xk=1.122741
The root of your polynomial is : 1.122741
Number of Repetitions:3
The total cost of the Newton-Rapshon method is:
Number of Additions and Substractions: 39
Number of Multiplications: 132
Number of Divisions: 11
```

Παρατηρούμε πως η λύση της μεθόδου είναι πάρα πολύ κοντά στην πραγματική λύση και σε μονάχα 3 επαναλήψεις. Αυτό συμβαίνει διότι, η αρχική μας εκτίμηση χ0=1 έτυχε να ήταν πολύ κοντά στην πραγματική λύση. Παρόλα αυτά δοκιμάστηκαν αρχικές εκτιμήσεις 10 φορές μικρότερη και 10 φορές μεγαλύτερη για να δούμε την αλλαγή στα αποτελέσματα όπως φαίνεται παρακάτω.

```
*******Newton-Raphson*******:
The first root estimation xo=0.100000
For repetition 0 :xk=10.441066
For repetition 1 :xk=8.308960
For repetition 2 :xk=6.605418
For repetition 3 :xk=5.245203
For repetition 4 :xk=4.160285
For repetition 5 :xk=3.296588
For repetition 6 :xk=2.611636
For repetition 7 :xk=2.073342
For repetition 8 :xk=1.660415
For repetition 9 :xk=1.364748
For repetition 10 :xk=1.190749
For repetition 11 :xk=1.129591
For repetition 12 :xk=1.122814
The root of your polynomial is : 1.122814
Number of Repetitions:13
The total cost of the Newton-Rapshon method is:
Number of Additions and Substractions: 169
Number of Multiplications: 572
Number of Divisions: 21
```

```
*******Newton-Raphson*******:
The first root estimation xo=10.000000
For repetition 0 :xk=7.956481
For repetition 1 :xk=6.323891
For repetition 2 :xk=5.020541
For repetition 3 :xk=3.981267
For repetition 4 :xk=3.154338
For repetition 5 :xk=2.499298
For repetition 6 :xk=1.986009
For repetition 7 :xk=1.595450
For repetition 8 :xk=1.322264
For repetition 9 :xk=1.171595
For repetition 10 :xk=1.126381
For repetition 11 :xk=1.122759
The root of your polynomial is : 1.122759
Number of Repetitions:12
The total cost of the Newton-Rapshon method is:
Number of Additions and Substractions: 156
Number of Multiplications: 528
Number of Divisions: 20
```

Παρατηρούμε ξανά πως ακόμα και για χ0=0.1 ή χ0=10 πως η λύση είναι επίσης πάρα πολύ κοντά στην πραγματική (ελάχιστα πιο μακριά από πριν), απλά χρειάστηκαν αρκετές παραπάνω επαναλήψεις μέχρι να επιτευχθεί συγκεκριμένα 13 και 12. Επομένως για περισσότερες επαναλήψεις είναι φανερό πως θα έχουμε και πολύ μεγαλύτερο κόστος πράξεων.

4 Μέθοδος Secant:

Η μέθοδος Secant ή αλλιώς η μέθοδος της τέμνουσας, είναι μια προσεγγιστική μέθοδος της Newton-Raphson , με ένα μεγάλο πλεονέκτημα. Το πλεονέκτημα της μεθόδου secant είναι πως δεν χρειάζεται η γνώση της παραγώγου f'(x) , καθώς και την υπολογίζει προσεγγιστικά με την απόσταση 2 σημείων να διαιρεί την απόσταση των τιμών αυτών των σημείων dy/dx. Αναλυτικότερα, επιλέγουμε ένα πολύ μικρό διάστημα μετατόπισης δ του σημείου χ και παίρνουμε τους τύπους:

$$\boldsymbol{dx = x_n * (1+\delta) - x_n \quad \kappa\alpha\iota \quad dy = f\big(x_n * (1+\delta)\big) - f(x_n)}$$

Έπειτα καταλήγουμε στον τύπο της ρίζας της πρώτης προσέγγισης:

$$\boldsymbol{x_1 = x_0 - \frac{f(x_0)}{\frac{dy}{dx}|x_0} \; (3)}$$

Επομένως, η γενική αναδρομική σχέση της μεθόδου είναι:

$$\boldsymbol{x_{n+1} = x_n - \frac{f(x_n)}{\frac{dy}{dx}|x_n} \; (4)}$$

Η υλοποίηση της μεθόδου ήταν ακριβώς ίδια με την Newton-Raphson , απλά με υλοποιημένος τους τύπους διαφορετικά χωρίς την χρήση της συνάρτησης υπολογισμού της παραγώγου. Το αποτέλεσμα της εξόδου είναι αντίστοιχο ,όπως και όλοι οι αντίστοιχοι υπολογισμοί με διαφορά στο κόστος τώρα δημιουργίας των τύπων.

Παρακάτω φαίνεται μια φωτογραφία ενός αποτελέσματος για το πολυώνυμο που δείχθηκε νωρίτερα:

```
***********Secant***********:
The first root estimation xo=1.000000
For repetition 0 :xk=1.153563
For repetition 1 :xk=1.124284
The root of your polynomial is : 1.124284
Number of Repetitions:2
The total cost of the Secant method is:
Number of Additions and Substractions: 40
Number of Multiplications: 122
Number of Divisions: 4
```

Παρατηρούμε πως το αποτέλεσμα είναι αρκετά κοντά στην λύση με μονάχα 2 επαναλήψεις (ισχύει ότι ίσχυε και πριν για την πρώτη εκτίμηση) και με ένα μικρό κόστος. Στην μέθοδο αυτή βέβαια εκτός από την εκτίμηση της αρχική λύσης υπάρχει άλλος ένας καταλυτικός παράγοντας που επηρεάζει την σύγκλιση και αυτός είναι η επιλογή του δ. Στο πρόγραμμα που υλοποίησα , παρατήρησα με δοκιμές πως τα αποτελέσματα των πολυωνύμων για αποδεκτό σφάλμα 0,1% έβγαζαν τα βέλτιστα αποτελέσματα σε με $\boldsymbol{\delta = 10^{-3}}$ το οποίο χρησιμοποιείται κιόλας για την παραπάνω εικόνα. Παρόλα αυτά, πειραματίστηκα με πολλές τιμές του δ για να γίνει απόλυτα αντιληπτό το πως επηρεάζει την παράγωγο. Γνωρίζω πως για πολύ μεγάλα δ , η ρίζα μπορεί να ταλαντώνεται γύρω από την λύση και εν τέλη να χρειαστούν παραπάνω επαναλήψεις για να συγκλίνει ή ακόμα και να μην συγκλίνει καθόλου. Αντίστοιχα για κάποιο πολύ μικρό δ ότι ίσως η σύγκλιση αργήσει πάρα πολύ. Παρακάτω φαίνονται τα αποτελέσματα για τιμές του δ:

$\boldsymbol{\delta = 10^{-6}}$:

```
***********Secant***********:
The first root estimation xo=1.000000
For repetition 0 :xk=1.148148
For repetition 1 :xk=1.119136
The root of your polynomial is : 1.119136
Number of Repetitions:2
The total cost of the Secant method is:
Number of Additions and Substractions: 40
Number of Multiplications: 122
Number of Divisions: 4
```

$\boldsymbol{\delta = 1}$:

```
***********Secant***********:
The first root estimation xo=1.000000
For repetition 0 :xk=1.028571
For repetition 1 :xk=1.049249
For repetition 2 :xk=1.064748
For repetition 3 :xk=1.076624
For repetition 4 :xk=1.085863
For repetition 5 :xk=1.093128
For repetition 6 :xk=1.098886
The root of your polynomial is : 1.098886
Number of Repetitions:7
The total cost of the Secant method is:
Number of Additions and Substractions: 140
Number of Multiplications: 427
Number of Divisions: 14
```

$\boldsymbol{\delta = 10}$:

```
***********Secant***********:
The first root estimation xo=1.000000
For repetition 0 :xk=1.000074
For repetition 1 :xk=1.000148
For repetition 2 :xk=1.000222
For repetition 3 :xk=1.000296
For repetition 4 :xk=1.000371
For repetition 5 :xk=1.000444
For repetition 6 :xk=1.000518
For repetition 7 :xk=1.000592
For repetition 8 :xk=1.000666
For repetition 9 :xk=1.000740
For repetition 10 :xk=1.000813
For repetition 11 :xk=1.000887
For repetition 12 :xk=1.000960
For repetition 13 :xk=1.001034
For repetition 14 :xk=1.001107
For repetition 15 :xk=1.001181
For repetition 16 :xk=1.001254
For repetition 17 :xk=1.001327
For repetition 18 :xk=1.001400
For repetition 19 :xk=1.001473
For repetition 20 :xk=1.001546
For repetition 21 :xk=1.001619
For repetition 22 :xk=1.001692
For repetition 23 :xk=1.001765
For repetition 24 :xk=1.001838
For repetition 25 :xk=1.001911
You have reached the maximum number 25 of repetitions and the root did not converges
```

Παρατηρώ πως για μικρότερο $\boldsymbol{\delta = 10^{-6}}$ η ρίζα πάλι συγκλίνει στις 2 επαναλήψεις αλλά αυτήν την φορά σε μια λίγο χειρότερη πιο μακρινή τιμή από την πραγματική ρίζα .Για $\boldsymbol{\delta = 1}$ παρατηρώ πως συγκλίνει σε μια ακόμα χειρότερη τιμή και με πολύ παραπάνω επαναλήψεις δηλαδή με 6 επαναλήψεις. Τέλος για $\boldsymbol{\delta = 10}$ όπως και ήταν αναμενόμενο δεν προλαβαίνει καν να γίνει η σύγκλιση της ρίζας μέσα σε 25 επαναλήψεις , οπότε η μέθοδος βγάζει ανεπιτυχής έξοδο.

5 Σύγκριση αποτελεσμάτων μεθόδων:

Συγκρίνοντας τις δύο μεθόδους όπως και ήταν αναμενόμενο η μέθοδος secant, δίνει λύσεις , λιγότερο κοντινές στην πραγματική ρίζα, αφού υπολογίζει την παράγωγο με προσεγγιστικό τρόπο και όχι αναλυτικά. Παρόλα αυτά , οι λύσεις αυτές είναι εντός του επαρκούς αποδεκτού σφάλματος για την χρήση σε εργαλεία CAD. Αντιθέτως, όσον αφορά την ταχύτητα και το κόστος η μέθοδος Secant υπερτερεί της μεθόδου Newton-Raphson, καθώς και υπολογίζει την ρίζα με λιγότερες επαναλήψεις και σημαντικά μικρότερο κόστος στο πλήθος των διαιρέσεων, λίγο μικρότερο κόστος σε πολλαπλασιασμούς και παρόμοιο κόστος προσθαφαιρέσεων.