ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη Εργαλείων CAD για Σχεδίαση Ολοκληρωμένων Κυκλωμάτων (ΗΡΥ 419 - ΗΡΥ 608)

PROJECT 1 – ΜΕΘΟΔΟΣ NEWTON-RAPHSON & ΜΕΘΟΔΟΣ ΕΦΑΠΤΟΜΕΝΗΣ

ΑΝΑΦΟΡΑ

ΜΠΙΜΠΙΡΗΣ ΠΕΤΡΟΣ (Α.Μ.: 2019030135)

Εισαγωγή:

Στο παρόν project μελετώνται η μέθοδος Newton-Raphson (στο εξής N-R) και η μέθοδος της εφαπτομένης, όσον αφορά την σύγκλιση τους στην ρίζα πολυωνύμων $5^{ου}$ βαθμού.

Η (γρήγορη) εύρεση της ρίζας ενός πολυωνύμου είναι σημαντικό μέρος ενός εργαλείου CAD για σχεδίαση ολοκληρωμένων κυκλωμάτων, και χρησιμοποιείται π.χ. για να βρεθεί το σημείο ισορροπίας του συστήματος – η τάση σε κάποιο κόμβο όταν το αλγεβρικό άθροισμα των ρευμάτων είναι 0.

Υλοποίηση:

Ο κώδικας που υλοποιεί τις μεθόδους που μελετώνται και εκτελεί τις δοκιμές είναι γραμμένος σε C, βλ. find_roots.c. Χρησιμοποιούνται ορισμένες δομές και συναρτήσεις από την «βιβλιοθήκη» polynomial.h, η οποία δημιουργήθηκε για τους σκοπούς αυτού του project.* Για να γίνει compile χρειάζεται μόνο να τρέξουμε το make:

```
$ make
```

Χρήση του προγράμματος

Το πρόγραμμα καλείται από την γραμμή εντολών (command line) ως εξής:

```
$ ./find_roots.exe <coefficients> <options>
```

όπου <coefficients> οι συντελεστές του πολυωνύμου του οποίου η ρίζα θα υπολογιστεί, διαχωρισμένες με κενούς χαρακτήρες και <options> διάφορες επιλογές παραμετροποίησης (βλ. παρακάτω). Για παράδειγμα, για το πολυώνυμο $8x^5 + 3x^4 + 6x^3 + 2x^2 + 12$, η κλήση του προγράμματος θα ήταν η εξής:

```
$ ./find_roots.exe 8 3 5 2 0 12
```

Τα options που μπορεί κανείς να δώσει ως arguments στο πρόγραμμα είναι τα εξής:

- `-i <n>` ⇒ οι μέθοδοι θα εκτελέσουν το πολύ n επαναλήψεις
- `-d <n>` ⇒ το δ που χρησιμοποιεί η μέθοδος της εφαπτομένης ορίζεται σε $n \in \mathbb{R}$
- `-x <n>` ⇒ το x_0 που χρησιμοποιούν οι μέθοδοι ορίζεται σε n
- `-v <n>` ⇒ τυπώνεται το x_0 και το $f(x_0)$ σε κάθε επανάληψη

Για παράδειγμα, τα παρακάτω είναι έγκυρες κλήσεις του προγράμματος:

```
$ ./find_roots.exe 8 3 5 2 0 12 -x 2.5
$ ./find_roots.exe 8 3 5 2 0 12 -i 100 -d 0.1 -v
```

By default, το x_0 είναι 1, το δ είναι 0.001 και ο μέγιστος αριθμός επαναλήψεων είναι 30.

Αποτελέσματα - Συμπεράσματα

Τα πολυώνυμα με τα οποία ελέγχθηκε η λειτουργία του προγράμματος καθώς και τα αποτελέσματα των δοκιμών φαίνονται παρακάτω:

$$p_1(x) = 8x^5 + 3x^4 + 6x^3 + 2x^2 + 12$$

$$p_2(x) = 3x^5 - 124x^4 + 439x^3 + 1234x^2 - 10439x + 931678$$

$$p_3(x) = x^5 - 42x^4 + 861x^3 + 9534x^2 - 60696x + 149688$$

$$p_4(x) = x^5 - 42x^4 + x^3 - 40x^2 - 78x - 246$$

$$p_5(x) = x^5 - 1337x^4 + 1x^3 - 1335x^2 - 2668x + 8023$$

Polynomial	Root	x_0	N-R Method Result (iterations)	Tangent Results (iterations)
$p_1(x)$	~ 1	0	$-\infty$ (20)	-86,280 (20)
		0	$-\infty$(100000)	-1.063 (44)
		1	-1.063 (17)	-1.063 (17)
		10^7	115292.076 (20)	115292.107 (20)
		10^7	-1.063 (115)	-1.063 (128)
$p_2(x)$	~ 37	0	36.977 (12)	36.977 (12)
		1	36.977 (14)	36.977 (14)
		10^7	115300.322 (20)	115300.348 (20)
		10^7	36.977 (64)	36.977 (64)
$p_3(x)$	~ 5	0	5.004 (7)	5.004 (6)
		1	5.004 (6)	5.004 (6)

		10^7	115300.453 (20)	115300.480 (20)
		10^7	5.004 (69)	5.004 (69)
$p_4(x)$	*42*	0	-1.560 (20)	2.014 (20)
		0	41.999 (715)	41.999 (71)
		1	-3.611 (20)	-1.271 (20)
		1	41.999 (109)	41.999 (939)
		10^7	115300.454 (20)	115300.488 (20)
		10^7	41.999 (63)	41.999 (63)
$p_5(x)$	*1337*	0	-1.845 (20)	-3.076 (20)
		0	1337.000 (2226)	1337.000 (4130)
		1	-0.603 (20)	-3.156 (20)
		1	1337.000 (325)	1337.000 (18590)
		10^7	115557.574 (20)	115557.612 (20)
		10^7	1337.000 (49)	1337.000 (48)

Πίνακας 1

Από τα αποτελέσματα προκύπτουν τα εξής συμπεράσματα:

- Υπάρχουν περιπτώσεις (συνδυασμοί πολυωνύμων - x_0 – σκιασμένες στον παραπάνω πίνακα) στις οποίες **οι 20 επαναλήψεις δεν είναι αρκετές** για να βρεθεί ρίζα. Σε αυτές τις περιπτώσεις, **επαναλαμβάνεται η εκτέλεση** του προγράμματος με έναν αρκετά μεγάλο επιτρεπόμενο αριθμό επαναλήψεων **προκειμένου να διαπιστωθεί ο αριθμός επαναλήψεων που χρειάζεται η κάθε μέθοδος** για να βρει μια ικανοποιητική ρίζα.
- Ο **μικρότερος αριθμός επαναλήψεων** για κάθε πολυώνυμο (σκιασμένος στον παραπάνω πίνακα) φαίνεται να εξαρτάται από την **διαφορά της ρίζας με το x_0** (όσο πιο κοντά στην ρίζα είναι το x_0, τόσο λιγότερες επαναλήψεις χρειάζονται) καθώς και με την **τιμή της παραγώγου (ή την ευθεία της εφαπτομένης) στο x_0.**
  Μια «καλή» επιλογή x_0 μπορεί να μειώσει κατά πολύ τον αριθμό επαναλήψεων και στις δύο μεθόδους, ένα μια «κακή» επιλογή μπορεί να έχει ακόμα και καταστροφικά αποτελέσματα (βλ. παρακάτω). Παρ' όλο που ένα «καλό» x_0 είναι δύσκολο να προσδιοριστεί είναι αρκετά πιο εύκολο να αποφύγουμε τις «κακές» επιλογές, απλά αποφεύγοντας τις τιμές που μηδενίζουν την παράγωγο του εκάστοτε πολυωνύμου. Έτσι είναι εγγυημένο ότι οι μέθοδοι θα τερματίσουν, και με μία χαλάρωση του κριτηρίου αποδοχής κάποιας τιμής ως ρίζα, ο αριθμός επαναλήψεων μπορεί να «κρατηθεί» αρκετά χαμηλά.
- Παρατηρούμε ότι η μέθοδος Newton-Raphson δεν τερματίζει στην περίπτωση του $p_1(x)$ με $x_0 = 0$. Αυτό συμβαίνει διότι η μέθοδος αυτή βασίζεται στην τιμή της πρώτης παραγώγου του πολυωνύμου προκειμένου να πλησιάσει την ρίζα, και $p_1'(0) = 0$, γεγονός το οποίο «αποπροσανατολίζει» την μέθοδο και την οδηγεί σε infinite loop.
- Η **τιμή του δ** στην μέθοδο της εφαπτομένης δεν επηρεάζει κατά πολύ την σύγκλιση η τον ρυθμό της, αρκεί να είναι μικρότερη της μονάδας (**κατά προτίμηση ≤ 0.1**). Όσο **μεγαλύτερη**

είναι η τιμή του δ τόσο περισσότερες επαναλήψεις χρειάζονται για να συγκλίνει η μέθοδος (σε κατά τ' άλλα αμετάβλητες συνθήκες). Αυτό είναι πλήρως αναμενόμενο, καθώς για «μεγάλες» τιμές του δ η ευθεία της οποίας η εξίσωση υπολογίζεται δεν είναι πλέον εφαπτομένη, είναι τέμνουσα (δηλ. τέμνει το πολυώνυμο σε 2 σημεία), οπότε **η ακρίβεια μειώνεται**. Για τον λόγο αυτό δεν περιλαμβάνονται οι αντίστοιχες τιμές του δ στον πίνακα – σε όλες τις δοκιμές που φαίνονται **ήταν ίσο με $\mathbf{0.001}$**.

- Παρατηρούμε επίσης ότι ακόμα και σε πολυώνυμα που γύρω από την ρίζα τους αλλάζουν απότομα τιμές (όπως π.χ. τα $p_2(x),\ p_4(x), p_5(x)$, οι γραφικές παραστάσεις των οποίων είναι αρκετά παρόμοιες), το πρόγραμμα βρίσκει την ρίζα με αρκετή ακρίβεια. Αυτό οφείλεται κυρίως στην χρήση μεταβλητών τύπου double (και όχι float) για τους υπολογισμούς.
- Σε κάθε περίπτωση, οι προσεγγιστικές αυτές μέθοδοι είναι αρκετά πιο γρήγορες από τον «μαθηματικό» υπολογισμό της ρίζας του πολυωνύμου, καθώς κάθε επανάληψη αντιστοιχεί σε λιγότερες από 5 αριθμητικές πράξεις.

*Μεταξύ των λόγων για τους οποίους η υλοποίηση έγινε με αυτόν τον τρόπο παρ' όλο που δεν ήταν απαραίτητο για τους σκοπούς του project ήταν η εξοικείωση με αυτόν τον τρόπο οργάνωσης του κώδικα καθώς και τις αλλαγές που επιφέρει στο compilation process (βλ. Makefile) σε σχέση με το «απλό» .c αρχείο. Η συγγραφή και οι δοκιμές έγιναν σε περιβάλλον Linux και το linking γίνεται δυναμικά, ωστόσο στο παραδοτέο περιλαμβάνεται ένα (statically compiled και linked) Windows executable.