Πανεπιστημιο Θεσσαλιας

πολυτεχνικη σχολη
Τμημα Μηχανικων Η/Υ, Τηλεπικοινωνιων και Δικτυων

Εφαρμογή της μεθόδου VAS σε περιβάλλον iOS

Διπλωματική Εργασία

Κεχαγιάς Σπυρίδων

Επιβλέποντες Καθηγητές: Αλκιβιάδης Γ. Ακρίτας
Αναπληρωτής Καθηγητής

Δασκαλοπούλου Ασπασία
Επίκουρος Καθηγήτρια

Βόλος, 2012

Πανεπιστημιο Θεσσαλιας

πολυτεχνικη σχολη
Τμημα Μηχανικων Η/Υ, Τηλεπικοινωνιων και Δικτυων

Εφαρμογή της μεθόδου VAS σε περιβάλλον iOS

Διπλωματική Εργασία

Κεχαγιάς Σπυρίδων

Επιβλέποντες Καθηγητές: Αλκιβιάδης Γ. Ακρίτας
Αναπληρωτής Καθηγητής

Δασκαλοπούλου Ασπασία
Επίκουρος Καθηγήτρια

Εγκρίθηκε από τη διμελή εξεταστική επιτροπή το Φεβρουάριο 2012.

(Υπογραφή) (Υπογραφή)

....

Βόλος, 2012

(Υπογραφή)

.....

Κεχαγιάς Σπυρίδων
Διπλωματούχος Μηχανικός Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων Πανεπιστημίου Θεσσαλίας

Αφιέρωση

Στην οικογένειά μου και τους φίλους μου.

Ευχαριστίες

Καταρχάς, ευχαριστώ θερμά τον επιβλέποντα καθηγητή μου κύριο Αλκιβιάδη Ακρίτα για την εμπιστοσύνη που έδειξε στο πρόσωπό μου, την αμέριστη βοήθεια, καθοδήγηση και υποστήριξή του κατά την εκπόνηση της παρούσας διπλωματικής εργασίας και όχι μόνο, και για την άριστη συνεργασία που οδήγησε στο παρόν τελικό αποτέλεσμα.

Ευχαριστώ πολύ την καθηγήτρια Ασπασία Δασκαλοπούλου που δέχτηκε να αναλάβει την επίβλεψη της διπλωματικής εργασίας μου.

Ευχαριστώ το δημιουργό της βιβλιοθήκης Giac και υπολογιστικού συστήματος Xcas, Bernarde Parisse για την καίρια βοήθειά του που διευκόλυνε κατά πολύ την περάτωση της παρούσας εργασίας.

Θα ήθελα να ευχαριστήσω επίσης την οικογένειά μου για τη συμπαράσταση και υποστήριξή τους τόσο κατά την εκπόνηση της εργασίας όσο και κατά τη διάρκεια των σπουδών μου.

Τέλος, οφείλω ένα μεγάλο ευχαριστώ στους φίλους και συμφοιτητές Νικόλαο Μακρή, Αλέξανδρο Ασβεστόπουλο, Κυριάκο Τσακμαλή για τη βοήθειά και υποστήριξή τους τόσο κατά τη διάρκεια των σπουδών μου όσο και σε θέματα σχετικά με την παρούσα διπλωματική εργασία.

Σπυρίδων Κεχαγιάς
Βόλος, 2012

Περίληψη

Σκοπός αυτής της εργασίας είναι η παρουσίαση της iOS εφαρμογής "Polynomial Real Roots", η οποία έχει ως κύριο χαρακτηριστικό τη χρήση της μεθόδου των συνεχών κλασμάτων Vincent–Akritas-Strzebonski (VAS Continued Fractions method) για την απομόνωση των πραγματικών ριζών ενός πολυωνύμου με ακέραιους ή ρητούς συντελεστές.

Αρχικά παρουσιάζουμε βασικές έννοιες σχετικές με το ζήτημα της απομόνωσης των πραγματικών ριζών ενός πολυωνύμου και κάνουμε μια σύντομη ιστορική ανασκόπηση γύρω από το θέμα αυτό. Ακολουθεί περιγραφή της ίδιας της μεθόδου VAS και της εξελικτικής της πορείας που οδήγησε στην εκδοχή με τα συνεχή κλάσματα που υλοποιεί η εφαρμογή μας. Εξετάζουμε, επίσης, παράγοντες που επιδρούν σημαντικά στην απόδοση του αλγορίθμου VAS, ήτοι την εύρεση και χρήση ορίων στις τιμές των θετικών ριζών των πολυωνύμων.

Στη συνέχεια, περνώντας από το θεωρητικό σκέλος της παρούσας εργασίας σε αυτό που είναι σχετικό με τον προγραμματισμό, εξετάζουμε θέματα σχετικά με την υλοποίηση του αλγορίθμου σε περιβάλλον iOS. Παρουσιάζουμε συνοπτικά θέματα που αφορούν το λειτουργικό σύστημα iOS όπως το iOS SDK, το περιβάλλον ανάπτυξης iOS εφαρμογών (Xcode) και την αντικειμενοστραφή γλώσσα που χρησιμοποιείται στο iOS, Objective-C. Έμφαση δίνεται στην περιγραφή της μαθηματικής C++ βιβλιοθήκης Giac, την οποία χρησιμοποιούμε για την υλοποίηση του αλγορίθμου. Τέλος, παρουσιάζουμε αναλυτικά τις λειτουργίες και τη διεπαφή της εφαρμογής μας.

Λέξεις Κλειδιά:
απομόνωση πραγματικών ριζών, VAS Continued Fractions, iOS εφαρμογή, Giac

Abstract

The purpose of this work is to present the iOS application "Polynomial Real Roots", the main feature of which is the use of the Vincent-Akritas-Strzebonski continued fractions method (VAS Continued Fractions method) for the real root isolation of a polynomial with integer or rational coefficients.

Initially, we present basic concepts concerning the issue of the isolation of a polynomial's real roots and we present a historical overview in relation to this subject. There follows a presentation of the VAS method itself, as well as its evolutionary course which has led to the 'continued fractions' version of the algorithm that is implemented in our application. We also examine factors that significantly affect the efficiency of the VAS algorithm, namely finding and using bounds on the values of the positive roots of the polynomials.

Moving from the theoretical part of this thesis to the programming-related, we examine issues concerning the implementation of the algorithm in iOS environment. We briefly present iOS-related subjects like the iOS SDK, its development environment (Xcode) and the object-oriented programming language used in iOS, Objective-C. Emphasis is placed on the description of the mathematical C++ library Giac, which we use for the implementation of the algorithm. Finally, we thoroughly present the features and interface of our application.

Keywords:
real root isolation, VAS Continued Fractions, iOS application, Giac

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Εισαγωγή

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η παρουσίαση της iOS εφαρμογής "Polynomial Real Roots". Η εφαρμογή αυτή, που αναπτύχθηκε για συσκευές iPhone και iPad, χρησιμοποιεί ως μαθηματικό υπολογιστικό πυρήνα τη C++ βιβλιοθήκη Giac, η οποία είναι επίσης η βάση του συστήματος υπολογιστικής άλγεβρας (Computer Algebra System – CAS) Xcas. Κεντρικό χαρακτηριστικό της εφαρμογής είναι η υλοποίηση και χρήση της μεθόδου συνεχών κλασμάτων Vincent-Akritas-Strzebonski που αναπτύχθηκε από τους Akritas και Strzebonski και αποτελεί τη στιγμή της συγγραφής της παρούσας εργασίας την ταχύτερη μέθοδο απομόνωσης πραγματικών ριζών πολυωνύμων.

Στο πρώτο μέρος της εργασίας θα παρουσιάσουμε το θεωρητικό μαθηματικό υπόβαθρο της μεθόδου VAS-CF (Continued Fractions), ενώ στο δεύτερο σκέλος θα εξετάσουμε θέματα υλοποίησης, τόσο του αλγορίθμου σε περιβάλλον iOS όσο και της διεπαφής (interface) της εφαρμογής που δίνει στο χρήστη τη δυνατότητα να κάνει χρήση του αλγορίθμου VAS, καθώς και άλλων συναρτήσεων που παρέχει η βιβλιοθήκη Giac.

Ειδικότερα, στο πρώτο κεφάλαιο της εργασίας παρουσιάζουμε αρχικά βασικές έννοιες σχετικές με την απομόνωση πραγματικών ριζών πολυωνύμων καθώς και χρήσιμο ιστορικό και μαθηματικό υπόβαθρο. Βλέπουμε συνοπτικά διάφορες άλλες μεθόδους που χρησιμοποιήθηκαν (ή χρησιμοποιούνται ακόμη) για την απομόνωση πραγματικών ριζών. Στη συνέχεια παρουσιάζουμε την εξελικτική πορεία της μεθόδου VAS που έχει οδηγήσει στην τρέχουσα εκδοχή της, η οποία και παρουσιάζεται αναλυτικότερα. Τέλος, αναδεικνύουμε η επίδραση στην απόδοση του αλγορίθμου που είχαν (και εξακολουθούν να έχουν) οι τιμές των ορίων στις τιμές των θετικών ριζών των πολυωνύμων. Η επίδραση αυτή γίνεται φανερή με τη χρήση νέων μεθόδων για την εύρεση ορίων με καλύτερη ακρίβεια, τις οποίες και αναφέρουμε.

Στην αρχή του δεύτερου κεφαλαίου παρουσιάζονται συνοπτικά θέματα σχετικά με το περιβάλλον iOS όπως οι συσκευές iOS, το ίδιο το λειτουργικό σύστημα iOS, το iOS SDK (με την Objective-C, την αντικειμενοστραφή γλώσσα προγραμματισμού στην οποία αναπτύσσονται iOS εφαρμογές) καθώς και το περιβάλλον ανάπτυξης iOS εφαρμογών, Xcode και των σημα-

ντικότερων εκ των εργαλείων που προσφέρει. Ακολούθως εξετάζεται η C++ βιβλιοθήκη Giac, που αποτελεί τον υπολογιστικό πυρήνα τις εφαρμογής και η οποία προσφέρει συναρτήσεις τόσο για την υλοποίηση της μεθόδου VAS όσο και για άλλες λειτουργίες που προσφέρονται στο χρήστη από την εφαρμογή (όπως είναι, για παράδειγμα, η παραγοντοποίηση πολυωνύμων). Στο τέλος του κεφαλαίου παρουσιάζονται αναλυτικά οι προσφερόμενες λειτουργίες και η διεπαφή της εφαρμογής.

Στο τέλος της εργασίας μπορούν να βρεθούν δύο παραρτήματα. Το πρώτο περιέχει τα βήματα που ακολουθήθηκαν για το cross-compiling της βιβλιοθήκης Giac για τον επεξεργαστή ARM που χρησιμοποιούν οι συσκευές iPhone, το οποίο και αποτέλεσε ακανθώδες σημείο κατά την εκπόνηση της διπλωματικής εργασίας. Στο δεύτερο παράρτημα παρατίθεται ενδεικτικός κώδικας της εφαρμογής, τόσο από το κομμάτι της ανάπτυξης της διεπαφής της εφαρμογής όσο και από το κομμάτι της υλοποίησης του αλγορίθμου.

Κεφάλαιο 1

Ο Αλγόριθμος VAS

Αντικείμενο του πρώτου κεφαλαίου είναι η εξέταση του ιστορικού και μαθηματικού υπόβαθρου θεμάτων σχετικών με την απομόνωση πραγματικών ριζών πολυωνύμων καθώς και η παρουσίαση και συζήτηση της μεθόδου συνεχών κλασμάτων VAS.

1.1 Βασικές έννοιες & ιστορική ανασκόπηση

Στην ενότητα αυτή θα παρουσιάσουμε βασικές έννοιες σχετικές με την απομόνωση πραγματικών ριζών πολυωνύμων και θα εξετάσουμε την ιστορική πορεία διάφορων προσπαθειών προς αυτή την κατεύθυνση.

Απομόνωση (isolation) των πραγματικών ριζών μιας πολυωνυμικής εξίσωσης είναι η διαδικασία εύρεσης πραγματικών διαστημάτων, μη τεμνομένων μεταξύ τους, έτσι ώστε κάθε διάστημα να περιέχει ακριβώς μία πραγματική ρίζα, και κάθε πραγματική ρίζα να περιέχεται σε κάποιο διάστημα. **Προσέγγιση** (approximation), από την άλλη μεριά, είναι η διαδικασία σμίκρυνσης των διαστημάτων απομόνωσης των ριζών τόσο, όσο να προσεγγισθούν αυτές στην επιθυμητή ακρίβεια.

Σύμφωνα με τη «Γαλλική Μαθηματική Σχολή» του 19^{ου} αιώνα, τα προβλήματα της απομόνωσης και προσέγγισης ριζών αποτελούν υποπροβλήματα του γενικότερου ζητήματος της επίλυσης πολυωνυμικών εξισώσεων με ακέραιους συντελεστές. Επομένως, η επίλυση του γενικότερου προβλήματος ανάγεται πλέον στην επίλυση των δύο επιμέρους υποπροβλημάτων. Το πρώτο πρόβλημα, η απομόνωση των ριζών, ήταν και το σημαντικότερο και έτσι άρχισαν οι προσπάθειες των μαθηματικών της εποχής εκείνης για την επίλυσή του.

Το 1637 ο Γάλλος René Descartes παρουσίασε – γενικεύοντας τα συμπεράσματα του Ιταλού μαθηματικού Gerolamo Cardano [5] – τον «κανόνα των προσήμων» του [15] (Descartes' rule of signs), που μας δίνει ένα

άνω όριο στο πλήθος των θετικών ριζών ενός πολυωνύμου.[1] Στις αρχές του 19ου αιώνα εμφανίστηκαν τα δύο θεωρήματα των Γάλλων F.D. Budan (1807) και J.B.J. Fourier (1820) [4]. Τα θεωρήματα αυτά, αν και διαφορετικά, είναι ισοδύναμα[2] και μας δίνουν ένα άνω φράγμα στο πλήθος των πραγματικών ριζών μιας πολυωνυμικής εξίσωσης σε ένα δεδομένο ανοιχτό διάστημα.

Το 1829, ο Γάλλος μαθηματικός C. Sturm, βασιζόμενος στο θεώρημα του Fourier, παρουσίασε ένα θεώρημα που μας επιτρέπει να απομονώσουμε τις πραγματικές ρίζες ενός πολυωνύμου χρησιμοποιώντας τη μέθοδο της **διχοτόμησης** [5]. Ο Sturm ήταν ο πρώτος που έλυσε το πρόβλημα της απομόνωσης των πραγματικών ριζών και η μέθοδος του χρησιμοποιήθηκε ευρέως μέχρι τα τέλη της δεκαετίας του 1980, όταν και αναπτύχθηκε από τον Αλκιβιάδη Γ. Ακρίτα η μέθοδος συνεχών κλασμάτων VAS.

Το 1836 παρουσιάστηκε το θεώρημα του Vincent[3] [26], το οποίο εξαρτάται από τον κανόνα των προσήμων του Descartes, ενώ κανείς μπορεί να εξακριβώσει και τη συνάφεια του με το θεώρημα του Budan μελετώντας και συγκρίνοντας τα δύο θεωρήματα. Το θεώρημα αυτό αποτελεί τη βάση δύο μεθόδων απομόνωσης των πραγματικών ριζών πολυωνυμικών εξισώσεων με συνεχή κλάσματα. Η πρώτη από αυτές τις μεθόδους ανήκει στον ίδιο τον Vincent (1836) και έχει εκθετικό χρόνο υπολογισμού [26]. Η δεύτερη αναπτύχθηκε το 1978 από τον Αλκιβιάδη Γ. Ακρίτα [1], βελτιώθηκε με τον Strzebonski (VAS) το 1994 [9] και έχει πολυωνυμικό χρόνο υπολογισμού. Χρησιμοποιείται στο σύστημα υπολογιστικής άλγεβρας Mathematica.

Το 1975-1976 αναπτύχθηκε η μέθοδος Vincent-Collins-Akritas (VCA) από τους Collins και Ακρίτα [16] η οποία στηρίζεται σε μία ολική τροποποίηση του θεωρήματος του Vincent. Η μέθοδος αυτή χρησιμοποιεί διχοτόμηση για την απομόνωση των πραγματικών ριζών, γι' αυτό και είναι γνωστή ως VCA-bisection method. Χρησιμοποιείται στο σύστημα υπολογιστικής άλγεβρας Maple. Η ταχύτερη υλοποίηση της μεθόδου αναπτύχθηκε το 2004 από τους Rouillier και Zimmerman [16]. Μια δεύτερη μέθοδος διχοτόμησης βασιζόμενη στο θεώρημα του Vincent αναπτύχθηκε το 2000 από τους Alesina-Galuzzi [14] (Vincent-Alesina-Galuzzi bisection method).

---

[1] Διατύπωση του κανόνα αυτού μπορεί να βρεθεί στην επόμενη ενότητα.

[2] Εξαιτίας της μεγάλης σημασίας τους, δημιουργήθηκε έριδα σχετικά με το ποιο θεώρημα ανακαλύφθηκε πρώτο. Τα επόμενα χρόνια, ωστόσο, το θεώρημα του Budan επισκιάστηκε από αυτό του Fourier σε σημείο να είναι δυσεύρετη στη βιβλιογραφία η αρχική διατύπωση του.

[3] Διατύπωση του θεωρήματος μπορεί να βρεθεί στην επόμενη ενότητα.

Το 1978 αναπτύχθηκε, όπως ήδη αναφέραμε, η μέθοδος συνεχών κλασμάτων από τον Αλκιβιάδη Γ. Ακρίτα, η οποία χρησιμοποιεί συνεχή κλάσματα για την απομόνωση πραγματικών ριζών και η οποία αντικατέστησε τη μέθοδο του Sturm, όντας πολύ ταχύτερη. Το 1994 βελτιώθηκε από τον Strzebonski, ενώ η ταχύτερη υλοποίηση του αναπτύχθηκε το 2008 από τους Akritas, Strzebonski και Vigklas [13]. Η τελευταία εκδοχή της μεθόδου είναι αυτή που υλοποιήσαμε σε περιβάλλον iOS στα πλαίσια της παρούσας διπλωματικής εργασίας και αποτελεί την ταχύτερη μέθοδο απομόνωσης τη στιγμή της συγγραφής.

1.2 Μαθηματικό Υπόβαθρο

Στην ενότητα αυτή παρουσιάζουμε για λόγους πληρότητας και βαθύτερης κατανόησης της μεθόδου VAS-CF (η διατύπωση και περιγραφή της οποίας ακολουθεί σε επόμενη ενότητα) τον κανόνα των προσήμων του Descartes [15] και το θεώρημα του Vincent [26].

Ο κανόνας των προσήμων του Descartes.

Έστω $p(x)=c_n x^n + c_{n-1}x^{n-1} + \ldots + c_1 x + c_0$ μία πολυωνυμική εξίσωση βαθμού $n>0$ με πραγματικούς συντελεστές. Αν v είναι ο αριθμός των μεταβολών προσήμου στην ακολουθία των συντελεστών $\{c_n, c_{n-1}, \ldots, c_1, c_0\}$ – όπου οι μηδενικοί συντελεστές έχουν παραλειφθεί – και ρ_+ είναι ο αριθμός των θετικών ριζών της $p(x)=0$, τότε $\boldsymbol{v=\rho_+ + 2\lambda}$, όπου $\lambda \in \mathbb{Z}_{\geq 0}$.

Προσέχουμε πως ο κανόνας των προσήμων μας δίνει τον **ακριβή** αριθμό των θετικών ριζών **μόνο** στις περιπτώσεις που $v=0$ ή $v=1$ – οπότε κατ' ανάγκη $\lambda=0$.

Το θεώρημα του Vincent (1836).

Αν σε μία πολυωνυμική εξίσωση με ρητούς συντελεστές και χωρίς πολλαπλές ρίζες κάνουμε διαδοχικές αντικαταστάσεις της μορφής

$$x \leftarrow a_1 + \frac{1}{x}, x \leftarrow a_2 + \frac{1}{x}, x \leftarrow a_3 + \frac{1}{x}, \ldots$$

όπου $a_1 \geq 0$ είναι τυχαίος **μη αρνητικός** ακέραιος και $a_2, a_3, \ldots$ είναι τυχαίοι **θετικοί** ακέραιοι, $a_i > 0$, $i>1$, τότε το νέο πολυώνυμο που προκύπτει **είτε** δεν έχει καμία, **είτε** έχει μία μεταβολή προσήμου. Στην τελευταία περίπτωση η εξίσωση έχει ακριβώς μία θετική ρίζα που παρίσταται από το

συνεχές κλάσμα:

$$a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ddots}}}$$

ενώ στην πρώτη περίπτωση δεν υπάρχει θετική ρίζα.

Για την απομόνωση των αρνητικών ριζών, τις μετατρέπουμε πρώτα σε θετικές με την αντικατάσταση $x \leftarrow -x$ στο $p(x)$. Ο όρος να μην έχει το πολυώνυμο $p(x)$ πολλαπλές ρίζες δεν περιορίζει τη γενικότητα του θεωρήματος του Vincent, καθώς στην αντίθετη περίπτωση το διασπάμε σε παράγοντες ελεύθερους από τετράγωνα (square-free factorization) και απομονώνουμε τις ρίζες κάθε ενός από αυτούς.

Απομόνωση ριζών πολυωνύμων με το θεώρημα του Vincent

Χρησιμοποιώντας τις 2 ειδικές περιπτώσεις του κανόνα προσήμων του Descartes – 0 ή 1 μεταβολή προσήμου – το θεώρημα του Vincent μπορεί να χρησιμοποιηθεί για να απομονώσει τις θετικές ρίζες ενός δοσμένου πολυωνύμου $p(x)$.

Αν παραστήσουμε με το μετασχηματισμό Möbius $M(x) = \dfrac{ax+b}{cx+d}$ το συνεχές κλάσμα που οδηγεί σε ένα μετασχηματισμένο πολυώνυμο $f(x) = (cx+d)^n p\left(\dfrac{ax+b}{cx+d}\right)$ με μία μεταβολή προσήμου, τότε η μοναδική θετική ρίζα του $f(x)$ – στο διάστημα $]0,+\infty[$ – αντιστοιχεί στη θετική ρίζα του $p(x)$ που περιέχεται στο ανοιχτό διάστημα με άκρα τα $\dfrac{b}{d}$ και $\dfrac{a}{c}$. Τα άκρα αυτά αντιστοιχούν στο $M(0)$ και $M(\infty)$ αντίστοιχα.

Επομένως, για να απομονώσουμε τις θετικές ρίζες ενός πολυωνύμου, αρκεί να υπολογίσουμε, για *κάθε* ρίζα, τις μεταβλητές a, b, c, d του αντίστοιχου μετασχηματισμού Möbius $M(x) = \dfrac{ax+b}{cx+d}$ που οδηγεί στο μετασχηματισμένο πολυώνυμο $f(x) = (cx+d)^n p\left(\dfrac{ax+b}{cx+d}\right)$ με ***μία*** μεταβολή προσήμου.

1.3 Περιγραφή της μεθόδου VAS-CF

Στην ενότητα αυτή παρουσιάζουμε την κλασική μέθοδο συνεχών κλασμάτων των Vincent-Akritas-Strzebonski για τις θετικές ρίζες.

Η μέθοδος συνεχών κλασμάτων είναι μια άμεση υλοποίηση του θεωρήματος του Vincent και αποτελεί – με τον πολυωνυμικό χρόνο υπολογισμού της – βελτίωση της πρώτης μεθόδου συνεχών κλασμάτων που είχε αναπτύξει ο Vincent το 1836 [26] και η οποία έχει εκθετικό χρόνο υπολογισμού. Ο Vincent στη μέθοδό του υπολόγισε κάθε μερικό πηλίκο a_i ως ακολουθία μοναδιαίων προσαυξήσεων $a_i \leftarrow a_i + 1$, που είναι ισοδύναμες με αντικαταστάσεις της μορφής $x \leftarrow x+1$.

Από την άλλη μεριά, η μέθοδος συνεχών κλασμάτων VAS υπολογίζει κάθε μερικό πηλίκο a_i ως το κάτω όριο (lower bound), lb, στις τιμές των θετικών ριζών ενός πολυωνύμου – το λεγόμενο «ιδανικό» κάτω όριο θετικών ριζών, που υπολογίζει το ακέραιο μέρος της μικρότερης θετικής ρίζας [12]. Έτσι, στη μέθοδο VAS, τώρα θέτουμε $a_i \leftarrow lb$ ή, ισοδύναμα, κάνουμε την αντικατάσταση $x \leftarrow x + lb$, που κοστίζει σε χρόνο περίπου όσο και η αντικατάσταση $x \leftarrow x+1$. Περισσότερες πληροφορίες μπορούν να βρεθούν στα [2-4], [5 - Chapter 7].

Ακολουθεί ο αλγόριθμος για την απομόνωση των θετικών ριζών. Έστω το πολυώνυμο $f \in \mathbb{Z}[x] \setminus \{0\}$. Με $sv(f)$ υποδηλώνουμε τον αριθμό των μεταβολών προσήμου στην ακολουθία των μη μηδενικών συντελεστών του f. Για μη αρνητικούς ακεραίους a, b, c και d, τέτοιους ώστε $ad - bc \neq 0$, συμβολίζουμε το διάστημα με άκρα $\frac{b}{d}$ και $\frac{a}{c}$ (η διάταξη των οποίων δεν είναι γνωστή) ως $interval(a, b, c, d)$. Με τον όρο *interval data* υποδηλώνουμε μια λίστα της μορφής $\{a, b, c, d, p, v\}$ Η κλιμάκωση των ριζών στο 4^o βήμα του αλγορίθμου οφείλεται στον Strzebonksi (1994) [10], ενώ η τιμή της παραμέτρου lb_0 στο βήμα αυτό βρίσκεται εμπειρικά και στον αλγόριθμο VAS είναι ίση με 16.

Ο Αλγόριθμος συνεχών κλασμάτων VAS για τις θετικές ρίζες.

Είσοδος: $p(x) = 0$, μια πολυωνυμική εξίσωση με ακέραιους συντελεστές, χωρίς πολλαπλές ρίζες και $p(0) \neq 0$.

Έξοδος: Τα διαστήματα απομόνωσης των **θετικών** ριζών του $p(x)$ ή οι ακριβείς θετικές ρίζες σε μορφή διαστημάτων.

---

1. Αρχικοποιούμε τη λίστα των διαστημάτων απομόνωσης των ριζών *rootIsolationIntervals* $=\{\}$. Θέτουμε $f \leftarrow p(x)$ και υπολογίζουμε $v \leftarrow sv(f)$. Αν $v=0$ επιστρέφουμε την άδεια λίστα $\{\}$. Αν $v=1$ επιστρέφουμε τη λίστα $\{(0, ub)\}$, όπου ub είναι ένα άνω φράγμα στις θετικές ρίζες του f. Θέτουμε τη λίστα *interval data* $\{1, 0, 0, 1, f, v\}$ στη λίστα των διαστημάτων προς εξέταση *intervalsToBeProcessed*.

2. (* επεξεργασία διαστήματος (Βήματα #3-10) *)
Αν η λίστα *intervalsToBeProcessed* είναι άδεια, επιστρέφουμε τη λίστα *rootIsolationIntervals*, αλλιώς **επαναλαμβάνουμε** την εξής διαδικασία: βγάζουμε την **πρώτη** λίστα *interval data* $\{a, b, c, d, f, v\}$ από τη λίστα *intervalsToBeProcessed*.

3. Υπολογίζουμε ένα κάτω φράγμα lb στις θετικές ρίζες του f.

4. Αν $lb \geq lb_0$, θέτουμε $f(x) \leftarrow f(lb \cdot x)$, $a \leftarrow lb \cdot a$, $c \leftarrow lb \cdot c$ και $lb \leftarrow 1$.

5. Αν $lb \geq 1$, θέτουμε $f(x) \leftarrow f(x+lb)$, $b \leftarrow lb \cdot a + b$ και $d \leftarrow lb \cdot c + d$. Αν $f(0)=0$, επισυνάπτουμε το διάστημα $\left\{\frac{b}{d}, \frac{b}{d}\right\}$ στη λίστα *rootIsolationIntervals* και θέτουμε $f(x) \leftarrow \frac{f(x)}{x}$. Αν $v=0$, πηγαίνουμε στο Βήμα #2. Αν $v=1$, επισυνάπτουμε το διάστημα $interval(a, b, c, d)$ στη λίστα *rootIsolationIntervals* και πηγαίνουμε στο Βήμα #2.

6. Υπολογίζουμε $f_1(x) \leftarrow f(x+1)$ και θέτουμε $a_1 \leftarrow a$, $b_1 \leftarrow a+b$, $c_1 \leftarrow c$, $d_1 \leftarrow c+d$ και $r \leftarrow 0$. Αν $f_1(0)=0$, επισυνάπτουμε το διάστημα $\left\{\frac{b_1}{d_1}, \frac{b_1}{d_1}\right\}$ στη λίστα *rootIsolationIntervals* και θέτουμε $f_1(x) \leftarrow \frac{f_1(x)}{x}$ και $r \leftarrow 1$. Υπολογίζουμε $v_1 \leftarrow sv(f_1)$ και θέτουμε $v_2 \leftarrow v - v_1 - r$, $a_2 \leftarrow b$, $b_2 \leftarrow a+b$, $c_2 \leftarrow d$ και $d_2 \leftarrow c+d$.

7. Αν $v_2 > 1$, υπολογίζουμε $f_2(x) \leftarrow (x+1)^m f\left(\frac{1}{x+1}\right)$, όπου m είναι ο βαθμός του f. Αν $f_2(0)=0$, θέτουμε $f_2(x) \leftarrow \frac{f_2(x)}{x}$ και υπολογίζουμε $v_2 \leftarrow sv(f_2)$.

8. Αν $v_1 < v_2$, ανταλλάσσουμε (swap) το $\{a_1, b_1, c_1, d_1, f_1, v_1\}$ με το $\{a_2, b_2, c_2, d_2, f_2, v_2\}$.

9. Αν $v_1 = 0$, πηγαίνουμε στο βήμα 2. Αν $v_1 = 1$, επισυνάπτουμε το διάστημα $interval(a_1, b_1, c_1, d_1)$ στη λίστα *rootIsolationIntervals*, αλλιώς επισυνάπτουμε τη λίστα *interval data* $\{a_1, b_1, c_1, d_1, f_1, v_1\}$ στην **αρχή** της λίστας *intervalsToBeProcessed*.

10. Αν $v_2 = 0$, πηγαίνουμε στο βήμα 2. Αν $v_2 = 1$, επισυνάπτουμε το διάστημα $interval(a_2, b_2, c_2, d_2)$ στη λίστα *rootIsolationIntervals*, αλλιώς επισυνάπτουμε τη λίστα *interval data* $\{a_2, b_2, c_2, d_2, f_2, v_2\}$ στην **αρχή** της λίστας *intervalsToBeProcessed*. Πηγαίνουμε στο Βήμα #2.

Μία εναλλακτική αναδρομική περιγραφή του αλγορίθμου, όπως αυτή δόθηκε στο [7], φαίνεται στην Εικόνα 1.1.

The VAS continued fractions method

Input: A univariate, square-free polynomial $p(x) \in \mathbb{Z}[x]$, $p(0) \neq 0$,
and the Möbius transformation $M(x) = \frac{ax+b}{cx+d} = x,\ a, b, c, d \in \mathbb{Z}$
Output: A list of isolating intervals of the *positive* roots of $p(x)$;

1 $var \longleftarrow$ the number of sign changes of $p(x)$;
2 if $var = 0$ **then RETURN** $\emptyset$;
3 if $var = 1$ **then RETURN** $\{]a, b[\}$ //$a = \min(M(0), M(\infty))$,
$b = \max(M(0), M(\infty))$;
4 $b \longleftarrow$ a lower bound on the positive roots of $p(x)$;
5 if $\ell b > 1\{p \longleftarrow p(x + \ell b), M \longleftarrow M(x + \ell b)\}$;
6 $p_{01} \longleftarrow (x+1)^{deg(p)} p(\frac{1}{x+1})$, $M_{01} \longleftarrow M(\frac{1}{x+1})$//
Look for real roots in $]0, 1[$;
7 $m \longleftarrow M(1)$//Is 1 a root?;
8 $p_{1\infty} \longleftarrow p(x+1)$, $M_{1\infty} \longleftarrow M(x+1)$//Look for real roots in $]1, +\infty[$;
9 if $p(1) \neq 0$ then
10 | **RETURN** VAS$(p_{01}, M_{01}) \bigcup$ VAS$(p_{1\infty}, M_{1\infty})$
11 else
12 | **RETURN** VAS$(p_{01}, M_{01}) \bigcup \{[m, m]\} \bigcup$ VAS$(p_{1\infty}, M_{1\infty})$
13 end

Algorithm 1: The VAS(p, M) ''*continued fractions*'' algorithm, where the second argument is the Möbius transformation $M(x)$ associated with $p(x)$. For simplicity, Strzeboński's contribution is not included.

Εικόνα 1.1: Αναδρομική περιγραφή του αλγορίθμου VAS.

Ο παραπάνω αλγόριθμος εξασφαλίζει την απομόνωση των **θετικών** ριζών. Για την απομόνωση των **αρνητικών** ριζών πρώτα εξετάζουμε αν $p(x) = p(-x)$. Αν ισχύει η ισότητα, οι αρνητικές ρίζες είναι συμμετρικές με τις θετικές για τις οποίες έχουμε ήδη υπολογίσει τα διαστήματα απομόνωσης. Στην περίπτωση αυτή, λοιπόν, τα διαστήματα απομόνωσης των αρνητικών ριζών βρίσκονται στοιχειωδώς. Αν $p(x) \neq p(-x)$, τότε θέτουμε $p(x) \leftarrow p(-x)$, εκτελούμε τον παραπάνω αλγόριθμο ακόμη μία φορά και στο τέλος απεικονίζουμε τα διαστήματα απομόνωσης των ριζών στον αρνητικό ημιάξονα.

Όσον αφορά τις περιπτώσεις μηδενικών ριζών, εύκολα ελέγχουμε αν $p(0) = 0$ και αν η ισότητα ισχύει, θέτουμε $p(x) = \frac{p(x)}{x}$.

Απαιτήσεις σε μνήμη.

Κάνοντας χρήση κατάλληλης υπόθεσης, οι Akritas-Strzebonski έδειξαν πως ο αλγόριθμος μπορεί να δομηθεί με τέτοιο τρόπο, ώστε ο μέγιστος αριθμός των μετασχηματισμένων εκδοχών του πολυωνύμου που χρειάζεται να αποθηκευτούν ανά πάσα στιγμή να είναι το πολύ $1+log_2 n$, όπου n είναι ο βαθμός του αρχικού πολυωνύμου [10].

Σημειώνουμε πως ο αλγόριθμος χρησιμοποιεί τον κανόνα των προσήμων του Descartes ως έλεγχο τερματισμού (Βήμα #1) κι επίσης ότι εξαρτάται σε μεγάλο βαθμό από τον επαναλαμβανόμενο υπολογισμό των κάτω ορίων στις τιμές των θετικών ριζών των πολυωνύμων (Βήμα #3). Η ακρίβεια των ορίων αυτών επηρεάζει σε μεγάλο βαθμό την απόδοση του αλγορίθμου, για αυτό και υπήρξαν διάφορες προσπάθειες βελτίωσής τους, όπως θα δούμε και στην επόμενη ενότητα.

Κάνοντας χρήση της εύλογης υπόθεσης του «ιδανικού» κάτω ορίου στις τιμές των θετικών ριζών σε συνδυασμό με το γρήγορο αλγόριθμο μετασχηματισμού των von zur Gathen και Gerhard [18], ο χρόνος υπολογισμού της μεθόδου συνεχών κλασμάτων VAS είναι $O(n^4\tau^2)$ όπου n είναι ο βαθμός του πολυωνύμου και το τ φράσσει το πλήθος των bits των συντελεστών πολυωνύμου. Η υπόθεση αυτή αμφισβητήθηκε από πολλούς οι οποίοι υποστήριζαν πως δίχως αυτή, ο χρόνος υπολογισμού του αλγορίθμου θα ήταν εκθετικός. Το 2007, ο Sharma απέδειξε ότι, χωρίς καμία υπόθεση, ο χρόνος υπολογισμού της μεθόδου είναι $O(n^8\tau^3)$ [22, 23], κάτι που, παρ' όλα αυτά, προκαλεί έκπληξη, καθώς δε συνάδει με την πρακτική απόδοση της VAS (και για αυτό το λόγο ίσως χρειαστεί στο μέλλον να αποτελέσει αντικείμενο περαιτέρω έρευνας).

1.4 Βελτίωση της απόδοσης της μεθόδου VAS-CF

Στην ενότητα αυτή εξετάζουμε την επίδραση που έχουν στην απόδοση της μεθόδου VAS τα όρια στις τιμές των θετικών ριζών των πολυωνύμων και παρουσιάζουμε τις περιπτώσεις των υπαρχόντων ορίων.

Όπως αναφέρθηκε στην Ενότητα 1.3, η απόδοση της μεθόδου συνεχών κλασμάτων VAS εξαρτάται σε μεγάλο βαθμό από την ακρίβεια των κάτω ορίων στις τιμές των θετικών ριζών.

1.4.1 Θεωρητικό και ιστορικό υπόβαθρο ορίων

Η εύρεση ενός **κάτω** ορίου, *lb* (lower bound), στις τιμές των θετικών ριζών ενός πολυωνύμου $p(x)$, βαθμού n, συνίσταται αρχικά στον υπολογισμό ενός **άνω** ορίου, *ub* (upper bound), στις τιμές των θετικών ριζών του $x^n p\left(\frac{1}{x}\right)$ και στη συνέχεια στον υπολογισμό του ως $lb = \frac{1}{ub}$. Επομένως, είναι προφανής η ανάγκη εύρεσης μιας αποδοτικής μεθόδου για τον υπολογισμό των τιμών των θετικών ριζών πολυωνυμικών εξισώσεων.

Στην αρχική υλοποίηση της μεθόδου VAS, το 1978, τα κάτω όρια υπολογίζονταν χρησιμοποιώντας ένα θεώρημα του Cauchy (Cauchy bound) [20]. Το 1986 εμφανίστηκε το όριο του Κιουστελίδη (Kioustelidis) [19], το οποίο χρησιμοποιήθηκε στη SYNAPS υλοποίηση του VAS από τους Tsigaridas και Emiris το 2006 [25].

Το 2005, ο Ştefănescu με το θεώρημά του [24] βοηθά να αποκτηθεί μια βαθύτερη κατανόηση σχετικά με τη φύση των ορίων και εισάγει την ιδέα του «ταιριάσματος» ή «ζευγαρώματος» ενός θετικού συντελεστή πολυωνύμου με ένα «αζευγάρωτο» αρνητικό συντελεστή μικρότερου βαθμού[1]. Η εφαρμογή του θεωρήματος αυτού, ωστόσο, ήταν δυνατή μόνο για πολυώνυμα με άρτιο πλήθος μεταβολών προσήμου.

Το 2006, οι Akritas, Strzebonski και Vigklas γενίκευσαν το θεώρημα του Ştefănescu ώστε πια να εφαρμόζεται σε πολυώνυμα με οποιοδήποτε πλήθος μεταβολών προσήμου [11]. Αυτό έγινε εισάγοντας την ιδέα του διαχωρισμού ή «σπασίματος» ενός θετικού συντελεστή σε κομμάτια που θα «ζευγάρωναν» με αρνητικούς συντελεστές όρων με μικρότερο βαθμό [12]. Χρησιμοποιώντας το θεώρημα αυτό, αποκτήθηκαν νέα, τροποποιημένα όρια Cauchy και Kioustelidis τα οποία υπολογίζονται με «σπάσιμο» του συ-

[1] Για την ακρίβεια, το «ταίριασμα» γίνεται μεταξύ των *όρων* των πολυωνύμων.

ντελεστή του μεγιστοβάθμιου όρου και «ταίριασμα» των προκυπτόντων κομματιών με συντελεστές όρων μικρότερου βαθμού.

1.4.2 Όρια γραμμικής πολυπλοκότητας

Τα τροποποιημένα όρια των Cauchy (**C**) και Kioustelidis (**K**) είναι γραμμικής πολυπλοκότητας (linear complexity bounds).

Γενική ιδέα των ορίων γραμμικής πολυπλοκότητας:
Ο υπολογισμός των ορίων αυτών γίνεται ως εξής:

- *κάθε* αρνητικός συντελεστής του πολυωνύμου «ζευγαρώνει» με *έναν* από τους προηγούμενους «*αζευγάρωτους*» θετικούς συντελεστές.
- ως εκτίμηση του ορίου λαμβάνεται το μέγιστο από όλα τα υπολογισθέντα ριζικά.

Χρησιμοποιώντας το γενικευμένο θεώρημα για τα όρια, αναπτύχθηκε μια νέα μέθοδος γραμμικής πολυπλοκότητας για τον υπολογισμό άνω ορίων στις τιμές των θετικών ριζών πολυωνύμων με το όνομα ***first-λ*** **(FL)**, που εισήγαγε την ιδέα «ταιριάσματος» των πρώτων λ θετικών συντελεστών του πολυωνύμου, όπου λ είναι το πλήθος των αρνητικών συντελεστών [13].

Παρ' ότι τα υπολογισθέντα με τη μέθοδο first-λ όρια ήταν καλύτερα των Cauchy και Kioustelidis, υπήρχαν περιπτώσεις στις οποίες και τα τρία όρια αποτυγχάνουν παταγωδώς. Για το λόγο αυτό αναπτύχθηκε άλλη μία νέα μέθοδος γραμμικής πολυπλοκότητας για τον υπολογισμό άνω ορίων στις τιμές των θετικών ριζών πολυωνύμων με το όνομα ***local-max*** **(LM)**. Η μέθοδος LM «ψάχνει» για το μέγιστο θετικό συντελεστή σε μια ακολουθία θετικών συντελεστών και τον «σπάει» σε $\frac{c_{\max}}{2^t}$ κομμάτια, όπου το t υποδηλώνει το πόσες φορές έχει χρησιμοποιηθεί αυτός ο συντελεστής.

Μετά από εκτεταμένες δοκιμές διαπιστώθηκε ότι, ανάμεσα στα όρια γραμμικής πολυπλοκότητας το καλύτερο είναι το *min(FL,LM)* (ή αλλιώς *FL+LM)* δηλαδή το μικρότερο από τα όρια FL και LM [12]. Επίσης, συγκρίνοντας τους χρόνους υπολογισμού του VAS(Cauchy) με αυτούς του VAS(fl+lm) παρατηρήθηκε μια επιτάχυνση της τάξης του 15% [13].

1.4.3 Όρια τετραγωνικής πολυπλοκότητας

Για την περαιτέρω βελτίωση της απόδοσης της μεθόδου VAS-CF, αποφασίστηκε η χρήση ορίων τετραγωνικής πολυπλοκότητας στις τιμές των θετι-

κών ριζών με την ελπίδα ότι οι βελτιωμένες, πιο ακριβείς εκτιμήσεις τους θα αντιστάθμιζαν τον επιπλέον χρόνο που απαιτείται για τον υπολογισμό τους.

Γενική ιδέα των ορίων γραμμικής πολυπλοκότητας:
Ο υπολογισμός των ορίων αυτών γίνεται ως εξής:

- *κάθε* αρνητικός συντελεστής του πολυωνύμου «ζευγαρώνει» με *όλους* τους προηγούμενους θετικούς συντελεστές και λαμβάνεται το ελάχιστο των υπολογισθεισών τιμών.
- Ως εκτίμηση του ορίου λαμβάνεται το μέγιστο όλων αυτών των ελαχίστων.

Όπως φάνηκε από δοκιμές εκτέλεσης του VAS, σε γενικές γραμμές οι εκτιμήσεις των διαστημάτων απομόνωσης (κι επομένως και των ριζών) που ελήφθησαν με χρήση ορίων τετραγωνικής πολυπλοκότητας είναι πιο ακριβείς από τις αντίστοιχες που ελήφθησαν από όρια γραμμικής πολυπλοκότητας.

Οι Akritas, Strzebonski και Vigklas ανέπτυξαν τρία νέα όρια τετραγωνικής πολυπλοκότητας (*Cauchy Quadratic - CQ*, *First-λ Quadratic - FLQ*, *Local-Max Quadratic - LMQ*) επεκτείνοντας τα αντίστοιχα όρια γραμμικής πολυπλοκότητας Cauchy, first-λ και local-max [6], ενώ το όριο του Kioustelidis επέκτεινε ο Hong το 1998 (*Kioustelidis Quadratic [KQ]*) [17]. Δοκιμές έδειξαν ότι το όριο FLQ είναι ένα από τα καλύτερα και ταχύτερα όρια [8], ωστόσο, αποφασίστηκε η χρήση του LMQ στον VAS [13].

Στον Πίνακα 1.1 παρουσιάζονται ομαδοποιημένα τα όρια γραμμικής και τετραγωνικής πολυπλοκότητας.

Όρια

Γραμμική Πολυπλοκότητα	Τετραγωνική Πολυπλοκότητα
Cauchy (C)	Cauchy Quadratic (CQ)
Kioustelidis (K)	Kioustelidis Quadratic (KQ)
first-λ (FL)	first-λ Quadratic (FLQ)
Local-Max (LM)	Local Max Quadratic (LMQ)

Πίνακας 1.1: Όρια γραμμικής και τετραγωνικής πολυπλοκότητας.

1.4.4 Όριο Local-Max Quadratic (LMQ)

Το όριο τετραγωνικής πολυπλοκότητας Local-Max Quadratic είναι, όπως είπαμε, αυτό που έχει επιλεγεί να χρησιμοποιείται στον αλγόριθμο VAS και επομένως και στην υλοποίηση του στην εφαρμογή που αναπτύχθηκε στα πλαίσια της παρούσας εργασίας. Για λόγους πληρότητας και βαθύτερης κατανόησης, λοιπόν, θα προχωρήσουμε στην ενότητα αυτή σε μια αναλυτικότερη παρουσίαση του ορίου.

LMQ: "Local-Max" Quadratic

Για ένα πολυώνυμο $p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$, $(a_n > 0)$ με πραγματικούς συντελεστές και τουλάχιστον μία μεταβολή προσήμου, κάθε αρνητικός συντελεστής $a_i < 0$ «ζευγαρώνει» με *κάθε* έναν από τους προηγούμενους θετικούς συντελεστές a_j διαιρούμενο διά του 2^{t_j} – δηλαδή κάθε θετικός συντελεστής a_j «σπάει» σε *άνισα* κομμάτια, όπως γίνεται με *μόνο* τον τοπικά μέγιστο συντελεστή στο *local-max* όριο· το t_j τίθεται αρχικά στην τιμή 1 και προσαυξάνεται κάθε φορά που χρησιμοποιείται ο θετικός συντελεστής a_j – και λαμβάνεται το ελάχιστο πάνω σε όλα τα j· ακολούθως, λαμβάνεται το μέγιστο πάνω σε όλα τα i.

Δηλαδή έχουμε:

$$ub_{LMQ} = \max_{\{a_i<0\}} \min_{\{a_j>0:\, j>1\}} \sqrt[j-1]{-\frac{a_j}{\frac{a_j}{2^{t_j}}}}$$

Τη στιγμή της συγγραφής αυτής της εργασίας, το LMQ είναι θεωρητικά το ακριβέστερο όριο στις τιμές των θετικών ριζών ενός πολυωνύμου, και με τη χρησιμοποίησή του στον VAS έχει οδηγήσει σε συνολική επιτάχυνση κατά 40% σε σχέση με την αρχική του υλοποίηση [13].

Κεφάλαιο 2

Υλοποίηση της εφαρμογής Polynomial Real Roots

Αντικείμενο του δεύτερου κεφαλαίου είναι η παρουσίαση θεμάτων σχετικών με την υλοποίηση της εφαρμογής Polynomial Real Roots όπως το προγραμματιστικό περιβάλλον iOS, η μαθηματική C++ βιβλιοθήκη Giac, η διεπαφή της εφαρμογής κ.ά.

2.1 Περιβάλλον iOS

Στην ενότητα αυτή θα γίνει μια σύντομη συζήτηση γύρω από θέματα που αφορούν το iOS καθώς και μια συνοπτική παρουσίαση του προγραμματιστικού περιβάλλοντος.

2.1.1 Συσκευές iOS

Συσκευές iOS (iOS devices) είναι οι συσκευές που έχουν σχεδιαστεί από την Apple Inc. και τρέχουν το Unix-like λειτουργικό σύστημα iOS. Στις συσκευές αυτές περιλαμβάνονται τα iPhone, iPod touch και iPad. Οι συσκευές iOS λειτουργούν ως φορητοί media players και Internet clients, ενώ διαθέτουν και hardware χαρακτηριστικά όπως Retina Display, οθόνη πολλαπλής αφής, επιταχυνσιόμετρο (accelerometer), γυροσκόπιο (three-axis gyro). Στα βασικά hardware χαρακτηριστικά τους περιλαμβάνονται ισχυρός επεξεργαστής αρχιτεκτονικής ARM, μονάδα 3D γραφικών, GPS sensors, WiFi, Bluetooth κ.ά.

Η αναβάθμιση του υπάρχοντος λειτουργικού συστήματος είναι μια εύκολη διαδικασία που πραγματοποιείται μέσω iTunes. Η Apple αναβαθμίζει το hardware των προϊόντων της περιοδικά· κάθε νέο μοντέλο είναι γνωστό ως «γενιά». Μέχρι τη στιγμή της συγγραφής της παρούσας εργασίας έχουν υπάρξει πέντε γενιές iPhone, τέσσερις για το iPod touch και δύο για το iPad.

(α) iPhone 4s

(β) iPad 2

(γ) iPod Touch 4g

Εικόνα 2.1: iOS συσκευές

2.1.2 Το λειτουργικό σύστημα iOS

Το iOS είναι το Unix-like λειτουργικό σύστημα για τις φορητές συσκευές τις Apple, Inc. Τη στιγμή της συγγραφής αυτής της εργασίας (Νοέμβριος 2011), οι iOS εφαρμογές στο App Store – την πλατφόρμα συγκέντρωσης και διαμοίρασης όλων των iOS εφαρμογών – ξεπερνούν σε αριθμό τις 500.000, οι οποίες συνολικά έχουν μεταφορτωθεί πάνω από 18 δισεκατομμύρια φορές. Η τελευταία έκδοση του iOS (ως το Νοέμβριο του 2011) είναι το iOS 5.0.1 (βλ. Εικόνα 2.2).

Εικόνα 2.2:
iOS 5.0.1 σε iPhone 4s

Το *User Interface (UI)* του iOS, ή αλλιώς *διεπαφή χρήστη* βασίζεται στον άμεσο χειρισμό από το χρήστη και κάνει χρήση χειρονομιών πολλαπλής αφής (multi-touch gestures). Η αλληλεπίδραση με το χρήστη περιλαμβάνει χειρονομίες όπως *swipe*, *tap*, *pinch* και *reverse pinch*, ενώ μερικές εφαρμογές εκμεταλλεύονται εσωτερικά hardware χαρακτηριστικά των συσκευών όπως το επιταχυνσιόμετρο ή το γυροσκόπιο προβλέποντας ειδικές αντιδράσεις σε κινήσεις όπως την περιστροφή ή το απότομο κούνημα της συσκευής.

Στο iOS υπάρχουν 4 *abstraction layers* δηλαδή στρώσεις αφαίρεσης που «κρύβουν» τις λεπτομέρειες της υλοποίησης διαφόρων λειτουργιών του: το "Core OS layer", το "Core Services layer", το "Media layer", και το "Cocoa Touch layer" (Εικόνα 2.3).

Εικόνα 2.3:
iOS abstraction layers

2.1.3 Το iOS SDK

Το *iOS SDK* (Software Development Kit) είναι μια «εργαλειοθήκη» ανάπτυξης λογισμικού που δημιούργησε η Apple, Inc. για την ανάπτυξη λογισμικού για το iOS. Τρέχει πάνω σε Mac OS X και επιτρέπει στους προγραμματιστές τη δημιουργία εφαρμογών για τις iOS συσκευές, καθώς και τον έλεγχό τους σε προσομοιωτές των συσκευών αυτών (iPhone & iPad Simulator). Το περιβάλλον ανάπτυξης λογισμικού για το iOS είναι το Xcode, το οποίο θα εξετάσουμε πιο αναλυτικά στην επόμενη ενότητα. Η τελευταία έκδοση (έως το Νοέμβριο 2011) είναι το iOS 5 SDK, ενώ η τελευταία του Xcode είναι το Xcode 4.2.

Objective-C

Οι εφαρμογές για τις iOS συσκευές γράφονται στην αντικειμενοστραφή γλώσσα προγραμματισμού *Objective-C*, ενώ είναι εφικτός και ο προγραμματισμός σε C ή C++ για ορισμένα στοιχεία μιας εφαρμογής. Η Objective-C είναι αυστηρό υπερσύνολο της C· οποιοδήποτε C πρόγραμμα είναι δυνατό να μεταγλωττιστεί με έναν Objective-C μεταγλωττιστή κι επίσης είναι εφικτή η εισαγωγή C κώδικα μέσα σε μια Objective-C κλάση. Επίσης, είναι δυνατή η ανάμιξη και των τριών γλωσσών (C, C++ και Objective-C), με το συνδυασμό τους να είναι γνωστό ως Objective C++. Τη μέθοδο αυτή ακολουθήσαμε και στην ανάπτυξη της εφαρμογής μας. Ακολουθούν συνοπτικές περιγραφές ορισμένων από τα πιο σημαντικά χαρακτηριστικά της Objective-C.

- **Messages**

Το μοντέλο αντικειμενοστραφούς προγραμματισμού της Objective-C είναι βασισμένο στην αποστολή μηνυμάτων (message passing) σε στιγμιότυπα αντικειμένων. Η αποστολή μηνύματος αντικαθιστά τις κλήσεις μεθόδων/συναρτήσεων των άλλων γλωσσών προγραμματισμού.

- **Interfaces & implementations**

Η Objective-C απαιτεί η δήλωση (interface) και η υλοποίηση (implementation) μιας κλάσης να βρίσκονται σε χωριστά δηλωμένα μπλοκ κώδικα. Κατά σύμβαση, το interface τοποθετείται σε ένα header file, ενώ η υλοποίηση σε αρχείο κώδικα.

- **Protocols**

Τα πρωτόκολλα (protocols) δηλώνουν μεθόδους που μπορούν να υλοποιηθούν από οποιαδήποτε κλάση.

- **Properties**

Τα "properties" είναι χαρακτηριστικό της αναθεωρημένης έκδοσης της Objective-C με το όνομα Objective-C 2.0 και ουσιαστικά αποτελούν συντακτική διευκόλυνση για τη δήλωση και διαχείριση instance variables (μεταβλητών στιγμιοτύπου).

- **Reference Counting**

Επίσης χαρακτηριστικό της Objective-C 2.0 (και του iOS συγκεκριμένα), το "reference counting" είναι ένας μηχανισμός διαχείρισης μνήμης που χρησιμοποιεί μετρήσεις αναφορών σε αντικείμενα για την αποδέσμευσή τους όταν δεν υπάρχουν πια άλλες αναφορές σε αυτά.

Cocoa

Τα Cocoa και Cocoa Touch Frameworks αποτελούνται από βιβλιοθήκες, αντικειμενοστραφή APIs και runtimes και είναι πλήρως ενσωματωμένα στο Xcode. Ο συνδυασμός των εργαλείων του Xcode (όπως το Interface Builder) με την Objective-C και το Cocoa API επιτρέπουν την εύκολη και γρήγορη δημιουργία υψηλής ποιότητας εφαρμογών.

Άξιο αναφοράς είναι το MVC (Model-View-Controller) μοντέλο σχεδίασης που χρησιμοποιεί το Cocoa. Τα "models" ενθυλακώνουν τα δεδομένα της εφαρμογής, ευθύνη των "Views" είναι η απεικόνιση των δεδομένων αυτών ενώ οι "Controllers" λειτουργούν ως μεσολαβητές μεταξύ των δύο. Αυτός ο διαχωρισμός των καθηκόντων επιτρέπει να γίνονται ευκολότερα ο σχεδιασμός, η υλοποίηση και η συντήρηση των εφαρμογών. Ο Interface Builder, το εργαλείο σχεδίασης GUI των iOS εφαρμογών, κάνοντας χρήση του MVC μοντέλου, επιτρέπει στον προγραμματιστή να επικεντρωθεί στην «όψη» (View) μια εφαρμογής χωρίς να χρειάζεται συγγραφή επιπλέον κώδικα.

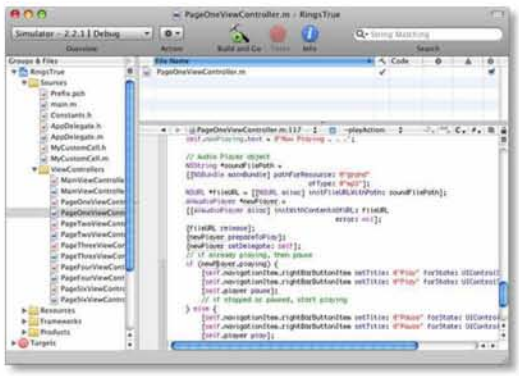
Εικόνα 2.4: MVC μοντέλο και Interface Builder.

2.1.4 Xcode

Το *Xcode* είναι μια σουίτα εργαλείων από την Apple, Inc. για την ανάπτυξη λογισμικού για τα Mac OS X και iOS. Αποτελεί, όπως προαναφέραμε, το περιβάλλον ανάπτυξης εφαρμογών για το iOS SDK. Η τελευταία έκδοση του Xcode τη στιγμής της συγγραφής (Νοέμβριος 2011) είναι το Xcode 4.2. Ακολούθως περιγράφουμε συνοπτικά μερικά από τα πιο σημαντικά εργαλεία που προσφέρει το Xcode για την ανάπτυξη λογισμικού.

Xcode IDE (Integrated Development Environment)

Το *Xcode IDE* αποτελεί το προγραμματιστικό περιβάλλον της σουίτας Xcode. Επιτρέπει τη συγγραφή κώδικα, οργάνωση των projects, διασύνδεση με άλλα κομμάτια του SDK και της σουίτας Xcode (όπως, π.χ. το documentation ή τις εφαρμογές Interface Builder / iOS Simulator), τη μεταγλώττιση κώδικα, τη φόρτωση εφαρμογών σε iOS συσκευές, το ανέβασμά τους στο App Store κ.ά.



(α) Xcode 3

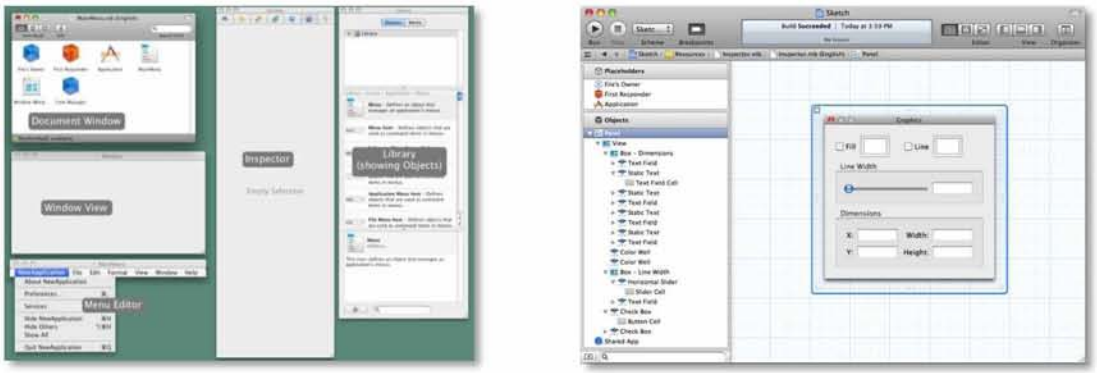
(β) Xcode 4

Εικόνα 2.5: Παραδείγματα Xcode IDEs διαφορετικών Xcode εκδόσεων.

Interface Builder (IB)

Ο *Interface Builder* είναι η εφαρμογή της σουίτας Xcode που επιτρέπει τη σχεδίαση του GUI (Graphical User Interface) μιας εφαρμογής. Η σχεδίαση του interface γίνεται με ένα απλό drag 'n' drop των αντικειμένων που επιλέγουμε σε κάποιο παράθυρο η μενού, ενώ προσφέρεται και η δυνατότητα σύνδεσης αντικειμένων-ενεργειών. Τα αντικείμενα επιλέγονται από μία βιβλιοθήκη αντικειμένων (*Library*) που προσφέρει ο Interface Builder, ενώ η επεξεργασία τους καθίσταται δυνατή μέσω του εργαλείου *Inspector*. Το in-

terface μιας εφαρμογής αποθηκεύεται από τον IB ως ένα "bundle" («δέμα» ομαδοποιημένων δεδομένων) που περιέχει τα αντικείμενα του interface και τις μεταξύ τους συνδέσεις. Τα αντικείμενα αυτά, στη συνέχεια, αρχειοθετούνται σε .xib (ή .nib) αρχεία.



(α) Xcode 3 (β) Xcode 4

Εικόνα 2.6: Παραδείγματα Interface Builders διαφορετικών Xcode εκδόσεων.

iOS Simulator

Ο *iOS Simulator* είναι ένας προσομοιωτής των συσκευών iPhone και iPad στον οποίον ο προγραμματιστής μπορεί να δοκιμάσει την εφαρμογή του. Προσφέρει τη δυνατότητα να δούμε αν το interface της εφαρμογής μας λειτουργεί σωστά, αν οι «όψεις» αλλάζουν όπως θα έπρεπε όταν η συσκευή περιστρέφεται, κι επίσης προσομοιώνει τις χειρονομίες αφής κάνοντας χρήση του ποντικιού.

(α) iPhone Simulator (β) iPad Simulator

Εικόνα 2.7: iPhone & iPad Simulator.

2.2 Η βιβλιοθήκη Giac

Στην ενότητα αυτή θα γίνει μνεία στη μαθηματική C++ βιβλιοθήκη Giac που χρησιμοποιήθηκε ως υπολογιστικός πυρήνας στην εφαρμογή μας.

Η *Giac* είναι μια δωρεάν (GPL) μαθηματική C++ βιβλιοθήκη που προσφέρει τύπους και συναρτήσεις για συμβολικές αλγεβρικές πράξεις. Αποτελεί τη βάση και τον υπολογιστικό πυρήνα του υπολογιστικού αλγεβρικού συστήματος γενικού σκοπού Xcas (που ουσιαστικά είναι ένα γραφικό interface – GUI – για τη βιβλιοθήκη Giac). Δημιουργός της βιβλιοθήκης και του Xcas είναι ο *Bernarde Parisse*. Περαιτέρω πληροφορίες σχετικά με τη βιβλιοθήκη, το Xcas ή το δημιουργό τους μπορούν να βρεθούν στην ιστοσελίδα του Xcas [29].

2.2.1 Μεταγλώττιση και εγκατάσταση της Giac

Η μεταγλώττιση της Giac ώστε να μπορεί να χρησιμοποιηθεί απρόσκοπτα τόσο στον desktop υπολογιστή όπου έγινε η ανάπτυξη της εφαρμογής όσο και σε iOS συσκευές ήταν μία από τις κύριες δυσκολίες που έπρεπε να αντιμετωπιστούν κατά την εκπόνηση αυτής της διπλωματικής εργασίας. Για τη δεύτερη διαδικασία ακολουθήθηκε μια διαδικασία γνωστή ως "cross-compiling", που σημαίνει τη δημιουργία εκτελέσιμου κώδικα για διαφορετική πλατφόρμα από αυτή στην οποία «τρέχει» ο μεταγλωττιστής. Το cross-compiling για iOS συσκευές συνίσταται στη δημιουργία κώδικα που θα μπορεί να εκτελεστεί σε επεξεργαστές αρχιτεκτονικής ARM, που είναι οι επεξεργαστές που χρησιμοποιούν οι iOS συσκευές. Τα βήματα που ακολουθήθηκαν για το cross-compiling της βιβλιοθήκης Giac αλλά και άλλων βιβλιοθηκών που είναι προαπαιτούμενες για τη μεταγλώττιση και εγκατάσταση της Giac παρουσιάζονται στο Παράρτημα Α της παρούσας εργασίας, ενώ εξετάζονται συνοπτικά και έννοιες σχετικές με το cross-compiling.

Για τη μεταγλώττιση, εγκατάσταση και χρησιμοποίηση της Giac είναι αναγκαίο να έχουν μεταγλωττιστεί και εγκατασταθεί προηγουμένως ορισμένες βιβλιοθήκες. Οι περισσότερες από αυτές είναι βοηθητικές βιβλιοθήκες (όπως οι 'libiconv', 'ncurses', 'gettext' και 'readline') που προσφέρουν περιφερειακές λειτουργίες, το πιο σημαντικό σημείο όμως είναι η ύπαρξη μιας βιβλιοθήκης αριθμητικής μεγάλης ακριβείας, είτε της *GMP* (**G**nu **M**ultiple **P**recision arithmetic library) είτε της *tommath*. Στα πρώτα στάδια ανάπτυξης της εφαρμογής μας γινόταν χρήση της GMP, ενώ στα τελικά

– όπως και στην τελική έκδοση της – χρησιμοποιήθηκε η tommath. Η τελευταία έκδοση της Giac αυτή τη χρονική στιγμή – η οποία και χρησιμοποιήθηκε στην εφαρμογή μας – είναι η 0.9.5 (unstable version).

Πέρα από τις προαπαιτούμενες αυτές βιβλιοθήκες, η Giac έχει τη δυνατότητα αξιοποίησης επιπλέον μαθηματικών βιβλιοθηκών, οι οποίες είναι προαιρετικές. Αυτές είναι οι MPFR, NTL, Pari, CoCoA, GSL και προσφέρουν, μεταξύ άλλων, προχωρημένες αριθμητικές συναρτήσεις, πράξεις πολυωνύμων και βελτίωση στην ταχύτητα σε ορισμένες λειτουργίες (π.χ. βάσεις Gröbner). Δε χρησιμοποιήθηκε κάποια από αυτές στην εφαρμογή μας.

2.2.2 Τύποι και συναρτήσεις της Giac

Η Giac χρησιμοποιεί τη γλώσσα C++, επειδή είναι ευκολότερο να γράψει κανείς αλγεβρικές πράξεις χρησιμοποιώντας τους συνηθισμένους τελεστές ('+', '-'. '*' κτλ.) αντί να χρησιμοποιήσει συναρτήσεις, κάτι που καθίσταται εφικτό στη C++ με την υπερφόρτωση τελεστών. Για παράδειγμα, η έκφραση `a+b*x` είναι ευκολότερο να κατανοηθεί και να τροποποιηθεί από την έκφραση `add(a,mul(b,x))`. Η C++ έχει, επίσης, χαρακτηριστικά και λειτουργίες που διευκολύνουν τον προγραμματισμό (σε σχέση με μια C βιβλιοθήκη, για παράδειγμα) όπως είναι τα ρεύματα εισόδου/εξόδου (I/O Streams) και η Standard Template Library (STL).

Η κλάση gen

Η *gen* είναι η γενική κλάση στην Giac που παριστάνει μαθηματικά αντικείμενα (`#include <giac/gen.h>`). Πρόκειται περί μιας C ένωσης (union) που αποτελείται είτε από «άμεσα» αντικείμενα (`int`, `double`) είτε από δείκτες σε πιο περίπλοκα αντικείμενα (λίστες/διανύσματα, ρητοί ή πραγματικοί αριθμοί κ.ά.). Ο πραγματικός τύπος μιας μεταβλητής τύπου gen, για παράδειγμα `gen e;`, μπορεί να εξακριβωθεί ελέγχοντας το πεδίο `type` (το οποίο είναι τύπου int), για παράδειγμα `if (e.type==...)`.

Μια μεταβλητή τύπου gen μπορεί, μεταξύ άλλων, να είναι:

1. Ακέραιος αριθμός int:
   - μπορεί να ελεγχθεί ως εξής: `e.type==_INT_`
   - λήψη τιμής γράφοντας: `int i=e.val;`
2. Αριθμός κινητής υποδιαστολής διπλής ακριβείας double:
   - μπορεί να ελεγχθεί ως εξής: `e.type==_DOUBLE_`
   - λήψη τιμής γράφοντας: `double d=e._DOUBLE_val;`
3. Ακέραιος μεγάλης ακριβείας:
   - μπορεί να ελεγχθεί ως εξής: `e.type==_ZINT`
   - λήψη τιμής γράφοντας: `mpz_t * m=e._ZINTptr;`

   Σημειώνουμε ότι ο τύπος `mpz_t` είναι τύπος της GMP· ο αντίστοιχος tommath τύπος είναι `mp_int`.
4. Αριθμός κινητής υποδιαστολής μεγάλης ακριβείας:
   - μπορεί να ελεγχθεί ως εξής: `e.type==_REAL`
5. Μιγαδικός αριθμός:
   - μπορεί να ελεγχθεί ως εξής: `e.type==_CINT`
   - ο αντίστοιχος δείκτης δείχνει σε 2 αντικείμενα τύπου gen: το πραγματικό μέρος gen `realpart=*e._CINTptr;` και το φανταστικό μέρος gen `imagpart=*(e._CINTptr+1);`.
6. Διάνυσμα (vector – στην ουσία είναι μια λίστα):
   - μπορεί να ελεγχθεί ως εξής: `e.type==_VECT`
   - ο δείκτης `e._VECTptr` δείχνει στον τύπο vecteur (συνώνυμο με το `std::vector<gen>`) . Για παράδειγμα η έκφραση `e._VECTptr->size()` θα δώσει το μέγεθος του αντικειμένου-διανύσματος.
7. Καθολικό όνομα/αναγνωριστικό:
   - μπορεί να ελεγχθεί ως εξής: `e.type==_IDNT`
   - ο αντίστοιχος δείκτης δείχνει στον τύπο identificateur: `identificateur i=*e._IDNTptr;`. Περισσότερες λεπτομέρειες μπορούν να βρεθούν στο αρχείο πηγαίου κώδικα giac/identificateur.h που έρχεται με το πακέτο της Giac.
8. Συμβολικό αντικείμενο:
   - μπορεί να ελεγχθεί ως εξής: `e.type==_SYMB`
   - ο αντίστοιχος δείκτης δείχνει στον τύπο symbolic: `symbolic s=*e._SYMBptr;`. Περισσότερες λεπτομέρειες μπορούν να βρεθούν στο αρχείο giac/symbolic.h.
9. Αλφαριθμητικό (string):
   - μπορεί να ελεγχθεί ως εξής: `e.type==_STRING`
     με αντίστοιχο δείκτη `*e._STRNGptr`.
10. Αντικείμενο-συνάρτηση:
    - μπορεί να ελεγχθεί ως εξής: `e.type==_FUNC`
    - ο αντίστοιχος δείκτης δείχνει στον τύπο unary_function_ptr: `unary_function_ptr u=*e._FUNCptr;`. Περισσότερες λεπτομέρειες μπορούν να βρεθούν στο αρχείο giac/unary.h.

Τα **πολυώνυμα** στην Giac μπορεί να είναι δύο τύπων:

- αραιά πολυώνυμα πολλών μεταβλητών, με όνομα τύπου `polynome`
- «πυκνά» (dense) πολυώνυμα μίας μεταβλητής, με όνομα τύπου `poly1` ή `modpoly`

Μια πλήρης λίστα των πιθανών τύπων μιας gen μεταβλητής και των αντίστοιχων δεικτών τους μπορεί να βρεθεί στο αρχείο `giac/dispatch.h`. Περισσότερες πληροφορίες σχετικά με τους τύπους γενικότερα και το πώς χρησιμοποιούνται στη βιβλιοθήκη υπάρχουν στην ιστοσελίδα του Xcas/Giac [29], μαζί με μικρά παραδείγματα C++ προγραμμάτων που χρησιμοποιούν την Giac.

Χρήση gen constructors

Για να δημιουργήσουμε κάποιο αντικείμενο τύπου gen θα πρέπει να κάνουμε χρήση των constructors της κλάσης gen. Οι δηλώσεις/πρότυπα των constructors αυτών βρίσκονται στο αρχείο `giac/gen.h`. Για παράδειγμα, για να αποθηκεύσουμε σε μια μεταβλητή τύπου gen με όνομα `poly1` το πολυώνυμο x^2-5x+6, ένας τρόπος θα ήταν ο εξής:

```
gen poly1("x^2-5x+6",0);
```

ή

```
gen poly1 = gen("x^2-5x+6",0);
```

Οι παραπάνω constructors δημιουργούν ένα αντικείμενο τύπου gen από αλφαριθμητικό. Η Giac διαθέτει ενσωματωμένο parser που αναγνωρίζει μαθηματικές εκφράσεις από αλφαριθμητικά ώστε να μπορούν να μετατραπούν ακολούθως σε αντικείμενα τύπου gen και να χρησιμοποιηθούν σε διάφορες πράξεις. Υπάρχουν πολλών ειδών constructors, που μπορούν να δημιουργήσουν αντικείμενα τύπου gen από τύπους `int`, `double`, `polynome`, `symbolic`, `identificateur`, `gen` κ.ά. Το δεύτερο όρισμα στις παραπάνω εντολές είναι μια σταθερά που ορίζει το context της Giac και την οποία θέτουμε ίση με '0'.

Κλήση συναρτήσεων

Με δεδομένη την έλλειψη πλήρους τεκμηρίωσης για τις συναρτήσεις της Giac, μπορεί κανείς να απευθυνθεί στο documentation του Xcas για να αναζητήσει πληροφορίες για αυτές, μιας και Giac και Xcas χρησιμοποιούν τις ίδιες συναρτήσεις με ελαφρώς παραλλαγμένα ονόματα.

Κατά κανόνα, λοιπόν, οι συναρτήσεις της Giac έχουν τη μορφή:

```
_nameOfXcasFunction(const gen &g, GIAC_CONTEXT),
```

όπου το 'nameOfXcasFunction' είναι, όπως είπώθηκε, το όνομα της συνάρτησης στο Xcas. Το δεύτερο όρισμα είναι ένα όρισμα-σταθερά που καθορίζει το context της Giac. Στην εφαρμογή μας το θέτουμε πάντα ίσο με '0'.

Όλες οι λειτουργίες στην Giac πραγματοποιούνται με χρήση του τύπου gen. Οι συναρτήσεις λαμβάνουν ως ορίσματα μεταβλητές τύπου gen (που μπορεί να είναι πολλών διαφορετικών υπο-τύπων όπως int, double, polynome, symbolic κτλ.) και επιστρέφουν επίσης gen.

Αν τα ορίσματα που πρέπει να περάσουμε σε μια συνάρτηση είναι περισσότερα από ένα, πρέπει πρώτα να τα τοποθετήσουμε σε μία λίστα (τύπος vecteur). Αυτό μπορεί να γίνει κάνοντας χρήση της συνάρτησης makevecteur() που παίρνει ως ορίσματα μία η παραπάνω μεταβλητές τύπου gen και επιστρέφει μια λίστα (vecteur) με στοιχεία τα ορίσματα αυτά. Σημειώνουμε πως η λίστα αυτή είναι και η ίδια γενικότερου τύπου gen.

Επομένως, αν θέλαμε να βρούμε το μέγιστο κοινό διαιρέτη των αριθμών 4 και 12, θα μπορούσαμε να ακολουθήσουμε την εξής διαδικασία (υπάρχουν φυσικά πολλοί εναλλακτικοί τρόποι για να επιτύχουμε το ίδιο αποτέλεσμα):

- Βρίσκουμε την αντίστοιχη συνάρτηση στο Xcas (μιας και εκεί το documentation είναι πλήρες). Στο παράδειγμα μας η συνάρτηση που ψάχνουμε είναι η gcd().
- Ψάχνουμε στον πηγαίο κώδικα που περιλαμβάνεται στο πακέτο της Giac για τη συνάρτηση _gcd(). Πράγματι, η συνάρτηση αυτή υπάρχει και περιλαμβάνεται στο αρχείο giac/usual.h. Το πρότυπο της συνάρτησης έχει ως εξής:

  ```
  gen _gcd(const gen & args,GIAC_CONTEXT);.
  ```

- Δημιουργούμε 2 μεταβλητές τύπου gen χρησιμοποιώντας C++ constructors (βλ. giac/gen.h), στις οποίες θα αποθηκεύσουμε τους ακεραίους 4 και 16.

  ```
  gen a = gen(4,0);      // type: int
  gen b = gen(16,0);     // type: int
  ```

- Στη συνέχεια τοποθετούμε τις μεταβλητές αυτές σε μια λίστα μέσω της συνάρτησης makevecteur() και περνάμε το αποτέλεσμα ως όρισμα στη συνάρτηση _gcd(). Τέλος, αποθηκεύουμε το επιστρεφό-

μενο αποτέλεσμα σε μια μεταβλητή τύπου gen με το όνομα result:

```
gen result = _gcd(makevecteur(a,b),0);
```

2.2.3 Η Giac στην εφαρμογή μας

Στην εφαρμογή Polynomial Real Roots, χρησιμοποιείται, όπως προαναφέρθηκε, η έκδοση 0.9.5 της Giac (unstable evolving version). Στην έκδοση αυτή, που δημοσιεύτηκε κατά το πέρας της παρούσας διπλωματικής εργασίας και ανάπτυξης της εφαρμογής, υπάρχει υλοποιημένη από το δημιουργό της Giac, Bernard Parisse, η private συνάρτηση *vas()* που καλείται από την public συνάρτηση *realroot()* και υλοποιεί την αντίστοιχη μέθοδο συνεχών κλασμάτων VAS. Στην εφαρμογή μας συνυπάρχουν αυτή και η δική μας υλοποίηση, εν τέλει επιλέξαμε όμως να χρησιμοποιούμε την πρώτη για λόγους απόδοσης μιας και αυτή η υλοποίηση αποδείχτηκε πιο συμβατή με τις –γρηγορότερες– εσωτερικές δομές της Giac κι επομένως ταχύτερη.

Πέραν, όμως, της VAS, η Giac χρησιμοποιείται στην εφαρμογή μας ως υπολογιστικός πυρήνας και για άλλες λειτουργίες και πράξεις που προσφέρονται στο χρήστη. Αυτές είναι:

1. Η απλή και square-free παραγοντοποίηση πολυωνύμων (συναρτήσεις `_factor()` και `_sqrfree()` αντίστοιχα).

2. Η απομόνωση ριζών κάνοντας χρήση της μεθόδου Sturm. Η μέθοδος Sturm υπάρχει υλοποιημένη στην Giac και καλείται από τη συνάρτηση `realroot()` δίνοντας ως όρισμα πριν το πολυώνυμο το αλφαριθμητικό "`sturm`".

3. Η εύρεση άνω και κάτω ορίων στις τιμές των θετικών ριζών πολυωνύμων. Σημειώνουμε πως, ενώ χρησιμοποιούνται οι συναρτήσεις της Giac για τον υπολογισμό τους, στην εφαρμογή συνυπάρχουν και οι δικές μας υλοποιήσεις που έγιναν στα πλαίσια της υλοποίησης της VAS.

4. Η εύρεση ρητών ριζών πολυωνύμων, που πραγματοποιείται κάνοντας χρήση της συνάρτησης `rationalroot()`.

Κάνοντας χρήση συναρτήσεων της Giac πραγματοποιήθηκε και η υλοποίηση της λειτουργίας της *προσέγγισης ριζών* πολυωνύμων (root approximation) που προσφέρει η εφαρμογή ως επιλογή στο χρήστη. Η συνάρτηση αυτή, όπως θα δούμε και σε επόμενη ενότητα όπου περιγράφουμε τα χα-

ρακτηριστικά της εφαρμογής, χρησιμοποιεί τη μέθοδο VAS, λαμβάνει τα προκύπτοντα διαστήματα απομόνωσης ως όρισμα και πραγματοποιεί προσέγγιση των ριζών σε δεδομένη ακρίβεια με διχοτόμηση.

Τέλος, παρ' όλο που η προσφερόμενη λειτουργία της σχεδίασης γραφικών παραστάσεων της εφαρμογής μας δεν οφείλεται στην Giac, ο parser της βιβλιοθήκης διευκολύνει σημαντικά τη σωστή αναγνώριση των πολυωνύμων που πρόκειται να σχεδιαστούν κάθε φορά.

Όλα τα προαναφερθέντα καθιστούν σαφή τη σημαντικότητα του ρόλου της βιβλιοθήκης στην ανάπτυξη του υπολογιστικού σκέλους της εφαρμογής μας.

2.3 Λειτουργίες και διεπαφή της εφαρμογής

Στην ενότητα αυτή θα παρουσιαστούν οι προσφερόμενες λειτουργίες καθώς και το interface της εφαρμογής Polynomial Real Roots.

2.3.1 Προσφερόμενες λειτουργίες

Το κύριο χαρακτηριστικό της εφαρμογής Polynomial Real Roots είναι βέβαια η υλοποίηση της μεθόδου VAS, αλλά δεν είναι το μοναδικό. Η εφαρμογή δίνει τη δυνατότητα στο χρήστη να επιλέξει μεταξύ διάφορων μαθηματικών λειτουργιών-πράξεων που επενεργούν πάνω σε ένα δοθέν πολυώνυμο. Στη συνέχεια παρουσιάζουμε τις λειτουργίες αυτές.

1. ***Απομόνωση πραγματικών ριζών*** πολυωνύμων με θετικούς συντελεστές ***κάνοντας χρήση της μεθόδου*** συνεχών κλασμάτων Vincent-Akritas-Strzebonski (***VAS***). Επιστρέφεται μια λίστα των διαστημάτων απομόνωσης μαζί με τις πολλαπλότητες των αντίστοιχων ριζών. Αν βρεθεί η ακριβής ρίζα m, τότε το διάστημα απομόνωσης που επιστρέφεται θα είναι της μορφής $[m,m]$. Η χρήση της μεθόδου VAS γίνεται, όπως αναφέρθηκε στην Ενότητα 2.2.3 με κλήση της Giac συνάρτησης `realroot()`.

2. ***Απομόνωση πραγματικών ριζών*** πολυωνύμων με θετικούς συντελεστές ***κάνοντας χρήση της μεθόδου Sturm***. Επιστρέφεται μια λίστα των διαστημάτων απομόνωσης μαζί με τις πολλαπλότητες των αντίστοιχων ριζών. Και σε αυτή την περίπτωση, αν βρεθεί η ακριβής ρίζα m, τότε το διάστημα απομόνωσης που ε-

πιστρέφεται θα είναι της μορφής $[m,m]$. Η χρήση της μεθόδου Sturm γίνεται, όπως αναφέρθηκε στην Ενότητα 2.2.3 με κλήση της Giac συνάρτησης `realroot()` βάζοντας ως πρώτο όρισμα το αλφαριθμητικό "sturm".

3. ***Προσέγγιση πραγματικών ριζών*** πολυωνύμων με θετικούς συντελεστές χρησιμοποιώντας τα διαστήματα απομόνωσης που επιστρέφονται από την κλήση της συνάρτησης `vas()` (που είναι η συνάρτηση που υλοποιεί τη μέθοδο VAS). Η προσέγγιση γίνεται με ***διχοτόμηση***. Επιστρέφεται μια λίστα με τις ρίζες προσεγγισθείσες στην ακρίβεια δεκαδικών ψηφίων που έχει θέσει ο χρήστης. Επιτρέπονται μέχρι 11 ψηφία ακρίβειας.

4. ***Σχεδίαση*** της ***γραφικής παράστασης*** της πολυωνυμικής εξίσωσης που αντιστοιχεί στο πολυώνυμο που δίνει ο χρήστης. Επιτρέπονται διάφορες ενέργειες - χειρονομίες αφής πάνω στη γραφική παράσταση όπως zoom in/out και κεντράρισμα στην αρχή των αξόνων, οι οποίες θα συζητηθούν εκτενέστερα στην επόμενη ενότητα που είναι σχετική με το interface της εφαρμογής.

5. ***Εύρεση κάτω και άνω ορίων*** στις θετικές ρίζες πολυωνύμων. Για τον υπολογισμό τους χρησιμοποιείται η Local Max Quadratic (LMQ – βλ. Ενότητα 1.4.4) μέθοδος. Η υλοποίηση στη Giac γίνεται με τις συναρτήσεις `poslbdLMQ()` και `posubLMQ()`.

6. ***Square-free factorization*** (διάσπαση πολυωνύμου σε παράγοντες ελεύθερους από τετράγωνα). Υλοποιείται με χρήση της Giac συνάρτησης `sqrfree()`.

7. ***Εύρεση των ρητών ριζών*** πολυωνύμων. Υλοποιείται με χρήση της Giac συνάρτησης `rationalroot()`.

8. Απλή ***παραγοντοποίηση*** πολυωνύμων. Η υλοποίηση γίνεται με τη συνάρτηση `factor()` της Giac.

Για κάθε μία από αυτές τις λειτουργίες υπάρχει στην εφαρμογή, όπως θα φανεί και στην επόμενη ενότητα, αντίστοιχο info button που εμφανίζει σχετικές πληροφορίες όταν πατηθεί. Επίσης, για τις λειτουργίες απομόνωσης ριζών με τις μεθόδους VAS και Sturm, πραγματοποιείται μέτρηση των χρόνων εκτέλεσής τους και εμφάνιση των αποτελεσμάτων στην οθόνη, ώστε να είναι εύκολη η σύγκριση της ταχύτητας των δύο μεθόδων.

2.3.2 Interface

Τα κύρια επιθυμητά χαρακτηριστικά που επιδιώξαμε για το interface (διεπαφή) της εφαρμογής μας είναι η απλότητα, η χρηστικότητα και η καλαισθησία. Το interface αποτελείται από δύο κεντρικές οθόνες στις οποίες προσφέρονται οι λειτουργίες που περιγράφηκαν στην Ενότητα 2.3.1. Στη συνέχεια περιγράφουμε τα αντικείμενα του interface της εφαρμογής και τη λειτουργία που επιτελούν, παραθέτοντας αντίστοιχες εικόνες.

Στην αρχική οθόνη της εφαρμογής υπάρχουν τα εξής αντικείμενα (συγκεντρωτικά εμφανίζονται στην Εικόνα 2.8):

α) Μια μπάρα με τον τίτλο της εφαρμογής "Polynomial Real Roots" στο πάνω μέρος της οθόνης, συνοδευόμενη από ένα info button που εμφανίζει γενικές πληροφορίες για αυτήν σε ένα alert window αν πατηθεί.

β) Text field με default κείμενο "Enter a polynomial here..." και συνοδευτικό info button με οδηγίες για τη σωστή εισαγωγή πολυωνύμων.

γ) Κουμπί για την απομόνωση ριζών του δοθέντος πολυωνύμου με τη μέθοδο VAS (βλ. λειτουργία #1 στην Ενότητα 2.3.1), συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.

δ) Κουμπί για την απομόνωση ριζών του δοθέντος πολυωνύμου με τη μέθοδο Sturm (βλ. λειτουργία #2 στην Ενότητα 2.3.1), συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.

ε) Κουμπί για την προσέγγιση των ριζών του δοθέντος πολυωνύμου με διχοτόμηση κάνοντας χρήση της μεθόδου VAS για την απόκτηση των διαστημάτων απομόνωσης των ριζών (βλ. λειτουργία #3 στην Ενότητα 2.3.1), συνοδευόμενο από ένα text field για την εισαγωγή των επιθυμητών ψηφίων ακρίβειας για την προσέγγιση και ένα info button για την εμφάνιση σχετικών πληροφοριών.

στ) Κουμπί για την σχεδίαση γραφικής παράστασης της πολυωνυμικής εξίσωσης που αντιστοιχεί στο δοθέν πολυώνυμο (βλ. λειτουργία #4 στην Ενότητα 2.3.1) συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών και επιτρεπόμενων gestures στην οθόνη της γραφικής παράστασης.

ζ) Κουμπί με τίτλο "More functions" που, αν πατηθεί, οδηγεί στη δεύτερη κεντρική οθόνη της εφαρμογής όπου ο χρήστης μπορεί να πραγματοποιήσει τις λειτουργίες #5, #6, #7 και #8 της Ενότητας 2.3.1.

Εικόνα 2.8: Η αρχική οθόνη της εφαρμογής Polynomial Real Roots.

Ακούμπώντας την οθόνη στο σημείο του text field, εμφανίζεται πληκτρολόγιο για να μπορέσει ο χρήστης να εισάγει πολυώνυμο (Εικόνα 2.9). Το πληκτρολόγιο εξαφανίζεται αν ο χρήστης ακουμπήσει στο background ή πατήσει "Done". Στην Εικόνα 2.10 απεικονίζεται ένα alert window που έχουμε προγραμματίσει να εμφανίζεται σε περίπτωση που έχει ζητηθεί κάποια ενέργεια ενώ έχει εισαχθεί μη έγκυρο πολυώνυμο. Στην Εικόνα 2.11 απεικονίζεται το alert window που εμφανίζεται όταν πατήσουμε το info button δεξιά του τίτλου της εφαρμογής μας.

Εικόνα 2.9:
Εισαγωγή πολυωνύμου μέσω πληκτρολογίου.

Εικόνα 2.10:
Εμφάνιση alert window λόγω μη έγκυρης εισόδου.

Εικόνα 2.11:
Εμφάνιση alert window με πληροφορίες.

Στις Εικόνες 2.12 και 2.13 απεικονίζονται ενδεικτικά οι οθόνες με τα αποτελέσματα που εμφανίζονται αν, με είσοδο το πολυώνυμο x^2-5x+6 (που έχει ρίζες τις '2' και '3'), πατηθούν τα κουμπιά "VAS Real Intervals", και "Sturm Real Intervals" αντίστοιχα. Εφιστούμε την προσοχή στους χρόνους εκτέλεσης που εμφανίζονται στο κάτω μέρος της οθόνης καθώς και στο κουμπί επιστροφής στην προηγούμενη (αρχική) οθόνη, πάνω αριστερά.

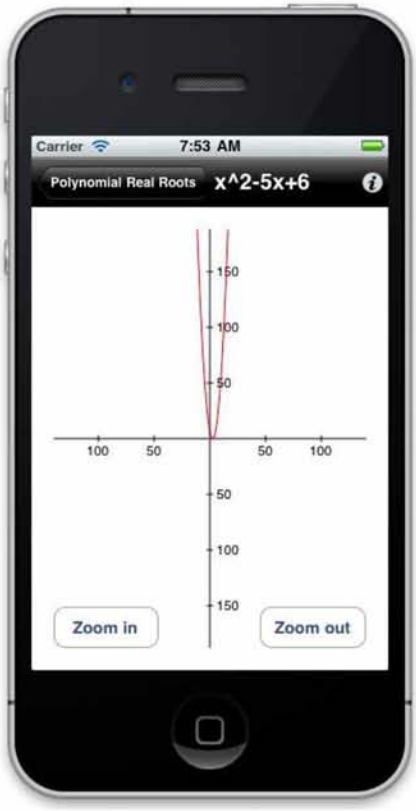
Εικόνα 2.12:
Η οθόνη με τα αποτελέσματα της μεθόδου VAS για το πολυώνυμο x^2+5x-6.

Εικόνα 2.13:
Η οθόνη με τα αποτελέσματα της μεθόδου Sturm για το πολυώνυμο x^2+5x-6.

Στις Εικόνες 2.14 και 2.15 απεικονίζονται ενδεικτικά οι αντίστοιχες οθόνες αν πατηθούν τα κουμπιά "Approximate Roots" (με precision digits ίσο με '6') και "Graph" αντίστοιχα. Να αναφέρουμε εδώ ότι στην οθόνη με τη γραφική παράσταση επιτρέπονται οι εξής χειρονομίες αφής (μπορούν να βρεθούν και μέσα στην εφαρμογή πατώντας το graph info button είτε στην αρχική οθόνη είτε στην οθόνη της γραφικής παράστασης):

- **Panning**: «Σέρνει» τη γραφική παράσταση προς την επιθυμητή κατεύθυνση – με ένα δάχτυλο.
- **Pinching**: Zooms in/out (διαθέσιμο επίσης μέσω κουμπιών).
- **Tapping twice**: Φέρνει την αρχή των αξόνων στο κέντρο της οθόνης.

Εικόνα 2.14:
Η οθόνη με τις προσεγγισθείσες ρίζες του πολυωνύμου x^2-5x+6 (ψηφία δεκαδικής ακρίβειας: 6).



Εικόνα 2.15:
Η οθόνη της γραφικής παράστασης για το πολυώνυμο x^2-5x+6.

Πατώντας το κουμπί "More Functions", ο χρήστης οδηγείται στη δεύτερη «κεντρική» σελίδα της εφαρμογής στην οποία υπάρχουν τα εξής αντικείμενα (συγκεντρωτικά εμφανίζονται στην Εικόνα 2.16):

α) Μια μπάρα με τον τίτλο "More Functions" στο πάνω μέρος της οθόνης και ένα κουμπί για την επιστροφή στην προηγούμενη (αρχική) οθόνη.

β) Text field που κρατά το πολυώνυμο που έχει δοθεί στην αρχική οθόνη (αν αυτό έχει δοθεί) και συνοδευτικό info button με οδηγίες για τη σωστή εισαγωγή πολυωνύμων.

γ) Κουμπί για την εύρεση κάτω ορίου στις τιμές των θετικών ριζών του δοθέντος πολυωνύμου. Ο υπολογισμός γίνεται με τη μέθοδο LMQ (βλ. λειτουργία #5 στην Ενότητα 2.3.1). Το κουμπί συνοδεύεται από ένα info button για την εμφάνιση σχετικών πληροφοριών.

δ) Κουμπί για την εύρεση άνω ορίου στις τιμές των θετικών ριζών του δοθέντος πολυωνύμου και συνοδευτικό info button για την εμφάνιση σχετικών πληροφοριών. Ο υπολογισμός γίνεται με τη μέθοδο LMQ (βλ. λειτουργία #5 στην Ενότητα 2.3.1).

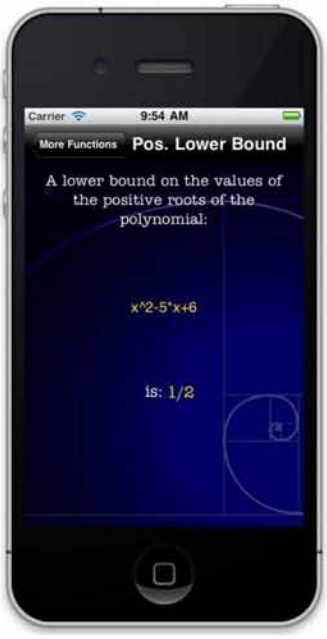
ε) Κουμπί για τη square-free παραγοντοποίηση του δοθέντος πολυωνύμου (βλ. λειτουργία #6 στην Ενότητα 2.3.1), συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.

στ) Κουμπί για την εύρεση των ρητών ριζών του δοθέντος πολυωνύμου (βλ. λειτουργία #7 στην Ενότητα 2.3.1) μαζί με info button για την εμφάνιση σχετικών πληροφοριών.

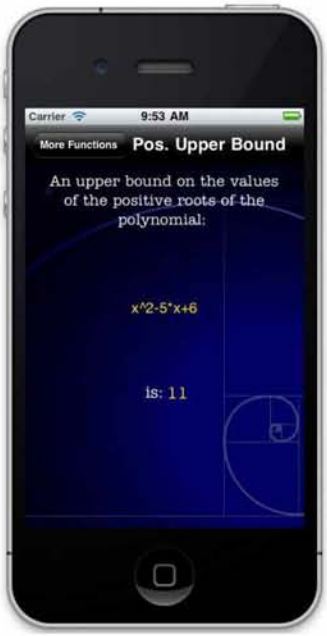
ζ) Κουμπί για την (απλή) παραγοντοποίηση του δοθέντος πολυωνύμου (βλ. λειτουργία #8 στην Ενότητα 2.3.1), συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.

Στις Εικόνες 2.17 και 2.18 απεικονίζονται ενδεικτικά οι οθόνες με τα αποτελέσματα που εμφανίζονται αν, με είσοδο το πολυώνυμο x^2-5x+6 (που έχει ρίζες τις '2' και '3'), πατηθούν τα κουμπιά "Positive Lower Bound", και "Positive Upper Bound" αντίστοιχα.

Εικόνα 2.16: Η οθόνη "More Functions" της εφαρμογής Polynomial Real Roots.



Εικόνα 2.17:
Η οθόνη με το κάτω όριο του πολυωνύμου x^2-5x+6.

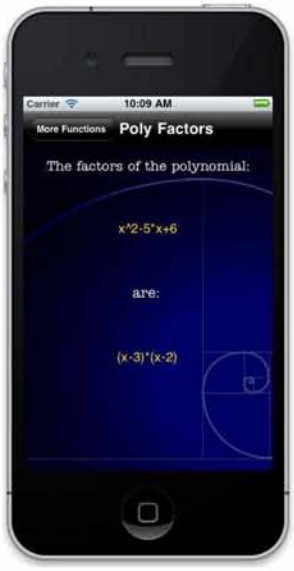


Εικόνα 2.18:
Η οθόνη με το άνω όριο του πολυωνύμου x^2-5x+6.

Στις Εικόνες 2.19, 2.20 και 2.21 απεικονίζονται ενδεικτικά οι αντίστοιχες οθόνες αν πατηθούν τα κουμπιά "Factor Square-Free", "Rational Roots" και "Factor" αντίστοιχα.

Εικόνα 2.19: Η οθόνη με τα αποτελέσματα της square-free παραγοντοποίησης για το πολυώνυμο x^2-5x+6.

Εικόνα 2.20: Η οθόνη με τις ρητές ρίζες του πολυωνύμου x^2-5x+6.



Εικόνα 2.21: Η οθόνη με τα αποτελέσματα της (απλής) παραγοντοποίησης για το πολυώνυμο x^2-5x+6.

Σε όλες τις παραπάνω απεικονίσεις, η συσκευή είναι κάθετα προσανατολισμένη (portrait). Η εφαρμογή μας υποστηρίζει και **οριζόντιο προσανατολισμό** (landscape). Παραδείγματα της εφαρμογής σε οριζόντια προσανατολισμένη συσκευή απεικονίζονται στην Εικόνα 2.22.

Εικόνα 2.22: Η εφαρμογή Polynomial Real Roots σε οριζόντια προσανατολισμένη συσκευή.

Συμπληρωματικά με όλες τις προαναφερθείσες λειτουργίες, η εφαρμογή μας παρέχει βοηθητικές λειτουργίες για τη διευκόλυνση του χρήστη. Ειδικότερα, έχει προβλεφθεί να διατηρείται το δοθέν πολυώνυμο στο input text field στις αλλαγές από οθόνη σε οθόνη και από προσανατολισμό σε προσανατολισμό (κάθετο σε οριζόντιο και το ανάποδο). Επίσης, κάθε φορά που ξεκινά να εκτελείται η εφαρμογή, ανακτάται και εμφανίζεται στο text field το τελευταίο πολυώνυμο που είχε εισάγει ο χρήστης κατά την προηγούμενη εκτέλεση. Έτσι, σε περιπτώσεις πιθανών κολλημάτων η σβησίματος της συσκευής, ο χρήστης δεν είναι αναγκασμένος να εισάγει πάλι το πολυώνυμο.

Εσωτερική δομή της εφαρμογής

Στην Ενότητα 2.1.3 αναφέραμε το σχεδιαστικό μοντέλο MVC (Model-View-Controller) που χρησιμοποιείται από το Cocoa στις iOS εφαρμογές. Πάνω στο μοντέλο αυτό βασίστηκε η εφαρμογή μας, όπως και όλες οι iOS εφαρμογές. Στη βάση της εφαρμογής μας βρίσκεται ένας "Root View Controller" που χειρίζεται τη δομή της εφαρμογής. Σε αυτόν τον "Root View Controller" τοποθετείται ως subview ένας "Navigation Root Controller" ο οποίος χειρίζεται μια στοίβα από άλλους Controllers και τα Views τους. Στην περίπτωση της εφαρμογής μας, καθώς πηγαίνουμε από τη μια οθόνη στην άλλη, αυτό που γίνεται δεν είναι τίποτα άλλο παρά το 'push' και το 'pop' τέτοιων Controllers στη στοίβα αυτή. Κάθε οθόνη, λοιπόν στην εφαρμογή μας αποτελεί την «όψη» (View) ενός Controller και κάθε φορά που μεταβαίνουμε από μια οθόνη σε μια άλλη, ο Controller αυτός εισάγεται στη στοίβα που αναφέραμε και εμφανίζει το αντίστοιχο View.

Παράρτημα Α

Μεταγλώττιση της βιβλιοθήκης Giac για iOS συσκευές

Αντικείμενο του Παραρτήματος Α είναι η εξέταση εννοιών σχετικών με το cross-compiling και η περιγραφή της διαδικασίας που ακολουθήθηκε για τη μεταγλώττιση της Giac για iOS συσκευές.

Cross-compiling είναι η δημιουργία εκτελέσιμου κώδικα για διαφορετική πλατφόρμα από αυτή στην οποία «τρέχει» ο μεταγλωττιστής. Το cross-compiling για iOS συσκευές σημαίνει τη δημιουργία κώδικα που θα μπορεί να εκτελεστεί στους επεξεργαστές που χρησιμοποιούν οι iOS συσκευές δηλαδή σε επεξεργαστές ARM.

Giac και προαπαιτούμενες βιβλιοθήκες

Για να μεταγλωττιστεί, εγκατασταθεί και χρησιμοποιηθεί η βιβλιοθήκη Giac πρέπει να έχουν μεταγλωττιστεί προηγουμένως και να υπάρχουν στο σύστημα ορισμένες βιβλιοθήκες. Οι περισσότερες από αυτές είναι βοηθητικές και προσθέτουν περιφερειακές λειτουργίες στην Giac, ωστόσο, είναι απαραίτητες για τη μεταγλώττισή της. Οι βιβλιοθήκες αυτές είναι οι εξής: 'libiconv', 'ncurses', 'gettext' και 'readline', ενώ ίσως χρειαστεί και η μεταγλώττιση της 'termcap'.

Απαραίτητη προϋπόθεση, επίσης, για τη μεταγλώττιση της Giac – είτε για την πλατφόρμα του συστήματος στο οποίο γίνεται η μεταγλώττιση είτε για κάποια άλλη πλατφόρμα – είναι η ύπαρξη στο σύστημα μιας μεταγλωττισμένης βιβλιοθήκης αριθμητικής μεγάλης ακριβείας. Αυτή μπορεί να είναι είτε η Gnu Multiple Precision arithmetic library (GMP) είτε η tommath. Στην τελική έκδοση της εφαρμογής μας χρησιμοποιήθηκε η tommath 0.39, αφού η GMP προκαλούσε αρκετά προβλήματα κατά τη μεταγλώττιση, ενώ η έκδοση της Giac που χρησιμοποιήθηκε είναι η 0.9.5 (unstable evolving version).

Όπως αναφέρθηκε στην ενότητα 2.2.1, η Giac έχει τη δυνατότητα αξιοποίησης και επιπλέον προαιρετικών μαθηματικών βιβλιοθηκών (MPFR, NTL, Pari, CoCoA, GSL), τις οποίες δε χρειάστηκε να χρησιμοποιήσουμε στην εφαρμογή μας κι επομένως δε μεταγλωττίστηκαν.

Shared (ή dynamic) και static βιβλιοθήκες

Συνήθως, όταν εγκαθιστούμε κάποια βιβλιοθήκη στο σύστημά μας (π.χ. Linux ή Mac OS X), παίρνουμε dynamic ή/και static βιβλιοθήκες. Dynamic ή shared βιβλιοθήκες ονομάζονται αυτές που φορτώνονται σε μια εφαρμογή τη στιγμή της φόρτωσης (load time) ή της εκτέλεσης (runtime) του προγράμματος, ενώ *static* είναι οι βιβλιοθήκες των οποίων οι υπορουτίνες και μεταβλητές φορτώνονται τη στιγμή της μεταγλώττισης (compile-time). Καθώς η χρήση δυναμικών βιβλιοθηκών σε εφαρμογές που τρέχουν σε iOS συσκευές αντενδείκνυται (το App Store απορρίπτει τέτοιες εφαρμογές), είναι αναγκαία η χρήση μόνο στατικών βιβλιοθηκών και αυτές θέλουμε να αποκτήσουμε με το cross-compiling. Οι static libraries είναι στην ουσία συλλογές από object files (με κατάληξη '.o') και συναντώνται κυρίως στη μορφή αρχείων με κατάληξη '.a'.

Η διαδικασία μεταγλώττισης της Giac (και των προαπαιτούμενων βιβλιοθηκών)

Το πρώτο βήμα που πρέπει να ακολουθήσουμε είναι το cross-compiling των περιφερειακών βιβλιοθηκών που προαναφέραμε και οι οποίες είναι απαραίτητες για τη μεταγλώττισης της Giac. Γενικά, η πλειοψηφία των βιβλιοθηκών περιέχουν στο πακέτο στο οποίο βρίσκονται διαθέσιμες για μεταφόρτωση από το Internet ένα αρχείο με οδηγίες εγκατάστασης (συνήθως με τον τίτλο 'INSTALL'), το οποίο περιέχει πληροφορίες για την εγκατάσταση της βιβλιοθήκης. Κατά πάσα πιθανότητα εκεί θα βρίσκονται και πληροφορίες για τυχόν εξαρτήσεις βιβλιοθηκών (library dependencies), αν υπάρχουν, από άλλες. Καλό είναι, επομένως, πριν την εγκατάσταση οποιασδήποτε βιβλιοθήκης να συμβουλεύεται ο κάθε ενδιαφερόμενος αυτές τις οδηγίες.

Στην περίπτωσή μας, γνωρίζοντας ότι η Giac είναι εξαρτώμενη από τις βιβλιοθήκες 'gettext' και 'readline', τις μεταφορτώσαμε στον υπολογιστή μας για να τις μεταγλωττίσουμε. Συμβουλευόμενοι τις οδηγίες εγκατάστασης της 'gettext', βλέπουμε, όμως, ότι και αυτή έχει δικές της εξαρτήσεις (ενώ η 'readline' είναι εξαρτώμενη της 'gettext'). Αυτές που χρειαζόμαστε, λοιπόν, για τη μεταγλώττιση της Giac είναι οι βιβλιοθήκες 'libiconv'

και 'ncurses' (ίσως χρειαστεί και η 'termcap' όπως θα δούμε ακολούθως). Οι διευθύνσεις για τη μεταφόρτωση των παραπάνω βιβλιοθηκών μπορούν να βρεθούν στον Πίνακα Α.1.

Πακέτα βιβλιοθηκών	Διεύθυνση μεταφόρτωσης
libiconv	ftp://ftp.gnu.org/gnu/libiconv/
ncurses	http://ftp.gnu.org/gnu/ncurses/
termcap	ftp://ftp.gnu.org/gnu/termcap/
gettext	http://ftp.gnu.org/gnu/gettext/
readline	ftp://ftp.gnu.org/gnu/readline/
tommath	1. http://libtom.org/?page=download&newsitems=5&whatfile=ltm
	2. http://www-fourier.ujf-grenoble.fr/~parisse/giac/tommath_iphone.tgz[1]
Giac	http://www-fourier.ujf-grenoble.fr/~parisse/giac_compile.html#source[2]

Πίνακας Α.1: Οι απαραίτητες βιβλιοθήκες για τη μεταγλώττιση της Giac και οι διευθύνσεις μεταφόρτωσής τους.

Αφού μεταφορτώσουμε όλες τις απαραίτητες βιβλιοθήκες, το επόμενο βήμα είναι το cross-compiling για κάθε μία από αυτές με τη σειρά που εμφανίζονται στον Πίνακα Α.1. Για το cross-compiling των βιβλιοθηκών χρησιμοποιήθηκε το script 'build_for_iphoneos' που περιέχεται μες στο πακέτο της Giac από την έκδοση 0.9.4 και μετά. Στο τέλος της ενότητας παρατίθεται το script αυτούσιο στη μορφή που υπάρχει μέσα στο πακέτο της Giac. Ενδεχόμενες αλλαγές στο script που είναι εξαρτώμενες από το εκάστοτε σύστημα στο οποίο γίνεται η μεταγλώττιση τονίζονται δίπλα από τον κώδικα με επεξηγηματικά σχήματα. Τα βήματα που ακολουθήθηκαν για τη μεταγλώττιση των βιβλιοθηκών είναι τα εξής:

Βήμα 1ο: Μεταγλώττιση της βιβλιοθήκης **libiconv**.

Έχοντας αποσυμπιέσει το πακέτο της βιβλιοθήκης libiconv, τοποθετούμε το script 'build_iphoneos' – που έχουμε πάρει μέσα από το αποσυμπιεσμένο πακέτο της Giac – μέσα στον αποσυμπιεσμένο φάκελο της libiconv. Ανοίγουμε ένα terminal window και μεταβαίνουμε (με cd) στο directory του

[1] Στη διεύθυνση αυτή περιέχεται η tommath 0.3.9 ήδη μεταγλωττισμένη από τον Bernard Parisse (ως static libraries) για Simulator και iPhone.
[2] Στην ίδια διεύθυνση μπορούν να βρεθούν τόσο το source package της Giac όσο και η βιβλιοθήκη μεταγλωττισμένη (ως static libraries) για Simulator και iPhone.

αποσυμπιεσμένου φακέλου (source) της libiconv. Πληκτρολογούμε την εξής εντολή στο terminal:

```
./build_for_iphoneos device --prefix=installationPath
```

όπου `installationPath` είναι το directory στο οποίο θέλουμε να εγκατασταθεί η βιβλιοθήκη. Στο ίδιο directory θα εγκαταστήσουμε και τις υπόλοιπες βιβλιοθήκες, οπότε δε θα πρέπει να μετακινηθούν οι παραχθείσες static libraries, καθώς οι εξαρτώμενες βιβλιοθήκες που θα μεταγλωττίσουμε στη συνέχεια θα περιμένουν να υπάρχουν οι προαπαιτούμενες τους εκεί. Οι static libraries της libiconv μαζί με τους αντίστοιχους headers θα πρέπει πια να βρίσκονται στο `installationPath`. Σημειώνουμε πως, στην περίπτωση που επιθυμούσαμε να μεταγλωττίσουμε τη βιβλιοθήκη για το Simulator του Xcode (δηλαδή για το σύστημα που γίνεται η μεταγλώττιση), θα έπρεπε να πληκτρολογήσουμε την εξής εντολή στο terminal, αντί για την παραπάνω:

```
./build_for_iphoneos simulator --prefix=installationPath
```

Βήμα 2ο: Μεταγλώττιση της βιβλιοθήκης **ncurses**.

Ομοίως με το Βήμα #1. Σε περίπτωση που κατά το compiling, ο compiler δεν μπορέσει να μεταγλωττίσει το termcap.h που περιέχεται στο πακέτο ncurses, θα πρέπει να μεταγλωττίσουμε τη βιβλιοθήκη termcap (Βήμα #3).

Βήμα 3ο: Μεταγλώττιση της βιβλιοθήκης **termcap**.

Ομοίως με το Βήμα #1. Στην περίπτωσή μας, επειδή χρησιμοποιήσαμε τον compiler 'llvm-gcc' (βλ. επεξηγηματικά σχήματα στο cross-compiling script στο τέλος της ενότητας) αντί για του 'gcc' που «περιμένει» να βρει η termcap, χρειάστηκε να αντικαταστήσουμε παντού μέσα στο αρχείο 'configure' που βρίσκεται μέσα στο source package της termcap, τη λέξη 'gcc' με 'llvm-gcc'. Ύστερα από αυτό, η μεταγλώττιση προχώρησε κανονικά.

Βήμα 5ο: Μεταγλώττιση της βιβλιοθήκης **gettext**.

Ομοίως με το Βήμα #1.

Βήμα 6ο: Μεταγλώττιση της βιβλιοθήκης **readline**.

Ομοίως με το Βήμα #1.

Βήμα 7ο: Μεταγλώττιση της βιβλιοθήκης **tommath**.

Εδώ δε χρειάστηκε να μεταγλωττίσουμε τη βιβλιοθήκη tommath, αφού υπήρχε ήδη μεταγλωττισμένη στη σελίδα του Xcas (βλ. δεύτερη διεύθυνση στην εγγραφή για την tommath στον Πίνακα Α.1). Χρησιμοποιήσαμε λοιπόν αυτή στην εφαρμογή μας. Τονίζουμε ότι τοποθετήσαμε τη βιβλιοθήκη και τα αντίστοιχα header files στο `installationPath` για να μπορέσει να τα βρει στη συνέχεια η Giac κατά τη μεταγλώττισή της.

Βήμα 8ο: Μεταγλώττιση της βιβλιοθήκης **Giac**.

Αν και θα μπορούσαμε να χρησιμοποιήσουμε αυτούσια τη μεταγλωττισμένη βιβλιοθήκη Giac που υπάρχει στη σελίδα του Xcas (βλ. εγγραφή για την Giac στον Πίνακα Α.1), επιλέξαμε να μεταγλωττίσουμε εκ νέου την τελευταία έκδοσή της (0.9.5). Η διαδικασία που ακολουθήσαμε είναι η ίδια με αυτή του Βήματος #1, με την εξαίρεση ότι η εντολή που πληκτρολογούμε στο terminal για την εκτέλεση του script είναι η εξής:

```
./build_for_iphoneos device --prefix=installationPath --enable-tommath
```

Η προσθήκη 'enable-tommath' στο τέλος της εντολής «λέει» στο script να γίνει η μεταγλώττιση της Giac κάνοντας χρήση της tommath (που έχουμε τοποθετήσει στο `installationPath`).

Με το πέρας του Βήματος #8, στο `installationPath` βρίσκονται πια όλες οι απαραίτητες βιβλιοθήκες μαζί με τα αντίστοιχα header files για να χρησιμοποιηθεί η Giac στο project της εφαρμογής μας.

Στη συνέχεια παραθέτουμε το cross-compiling script '`build_iphoneos`' συνοδευόμενο από σχήματα με επεξηγηματικά και άλλα σχόλια.

‘build_iphoneos’ original cross-compiling script

```
#!/bin/bash

###############################################################################
#
# Copyright (c) 2008-2009 Christopher J. Stawarz
#
# Permission is hereby granted, free of charge, to any person
# obtaining a copy of this software and associated documentation files
# (the "Software"), to deal in the Software without restriction,
# including without limitation the rights to use, copy, modify, merge,
# publish, distribute, sublicense, and/or sell copies of the Software,
# and to permit persons to whom the Software is furnished to do so,
# subject to the following conditions:
#
# The above copyright notice and this permission notice shall be
# included in all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
# EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
# MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
# NONINFRINGEMENT.  IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
# BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
# ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
# CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#
###############################################################################

# Disallow undefined variables
set -u

default_gcc_version=4.2
default_iphoneos_version=4.3
default_macos_version=10.6


GCC_VERSION="${GCC_VERSION:-$default_gcc_version}"
export
IPHONEOS_DEPLOYMENT_TARGET="${IPHONEOS_DEPLOYMENT_TARGET:-$default_iphoneos_version}"
export MACOSX_DEPLOYMENT_TARGET="${MACOSX_DEPLOYMENT_TARGET:-$default_macos_version}"

usage ()
{
    cat >&2 << EOF
Usage: ${0##*/} [-ht] [-p prefix] target [configure_args]
       -h     Print help message
       -p     Installation prefix (default: \$HOME/Developer/Platforms/...)
       -t     Use 16-bit Thumb instruction set (instead of 32-bit ARM)

The target must be "device" or "simulator".  Any additional arguments
are passed to configure.

The following environment variables affect the build process:

       GCC_VERSION                  (default: $default_gcc_version)
       IPHONEOS_DEPLOYMENT_TARGET   (default: $default_iphoneos_version)
       MACOSX_DEPLOYMENT_TARGET     (default: $default_macos_version)

EOF
}
```

Στην περίπτωσή μας αλλάξαμε το iOS από 4.3 σε 5.0.

Σε σύστημα με Mac OS X Lion ίσως χρειαστεί αλλαγή από 10.6 σε 10.7. Το δικό μας σύστημα έτρεχε Snow Leopard 10.6.7, οπότε δεν το αλλάξαμε.

```
while getopts ":hp:t" opt; do
    case $opt in
        h  ) usage ; exit 0 ;;
        p  ) prefix="$OPTARG" ;;
        t  ) thumb_opt=thumb ;;
        \? ) usage ; exit 2 ;;
    esac
done
shift $(( $OPTIND - 1 ))

if (( $# < 1 )); then
    usage
    exit 2
fi

target=$1
shift

case $target in
    device )
        arch=armv7
        platform=iPhoneOS
        extra_cflags="-m${thumb_opt:-no-thumb} -mthumb-interwork"
        ;;
    simulator )
        arch=i386
        platform=iPhoneSimulator
        extra_cflags="-
D__IPHONE_OS_VERSION_MIN_REQUIRED=${IPHONEOS_DEPLOYMENT_TARGET%%.*}0000"
        ;;
    * )
        usage
        exit 2
esac

platform_dir="/Developer/Platforms/${platform}.platform/Developer"
platform_bin_dir="${platform_dir}/usr/bin"
platform_sdk_dir="${platform_dir}/SDKs/${platform}${IPHONEOS_DEPLOYMENT_TARGET}.sdk"
prefix="${prefix:-${HOME}${platform_sdk_dir}}"

export CC="${platform_bin_dir}/gcc-${GCC_VERSION}"
export CFLAGS="-arch ${arch} -pipe -Os -gdwarf-2 -isysroot ${platform_sdk_dir} ${ex-
tra_cflags}"
export LDFLAGS="-arch ${arch} -isysroot ${platform_sdk_dir}"
export CXX="${platform_bin_dir}/g++-${GCC_VERSION}"
export CXXFLAGS="${CFLAGS}"
export CPP="/Developer/usr/bin/cpp-${GCC_VERSION}"
export CXXCPP="${CPP}"

./configure \
    --prefix="${prefix}" \
    --host="${arch}-apple-darwin" \
    --disable-shared \
    --enable-static \
    "$@" || exit

make install || exit

cat >&2 << EOF

Build succeeded!  Files were installed in

  $prefix

EOF
```

Οι αρχιτεκτονικές για τις οποίες επιλέγουμε να μεταγλωττίσουμε (armv7 για το iPhone, i386 για το Simulator) την εκάστοτε βιβλιοθήκη.

Στο νέο iOS SDK 5.0, που είχαμε στο σύστημά μας, default compiler είναι ο 'llvm'. Αντί για τον gcc υποστηρίζεται πια ο 'llvm-gcc'. Αλλάξαμε λοιπόν το 'gcc' σε 'llvm-gcc'.

Αλλάξαμε το 'g++' σε 'llvm-g++'.

Αλλάξαμε το 'cpp' σε 'llvm-cpp'.

Αλλάξαμε το 'make install' σε 'sudo make install' λόγω προβλημάτων με τα δικαιώματα.

Παράρτημα Β

Ενδεικτικός κώδικας της εφαρμογής Polynomial Real Roots

Στο Παράρτημα Β παραθέτουμε ενδεικτικά κομμάτια κώδικα από την υλοποίηση της εφαρμογής μας.

Για λόγους οικονομίας δεν παραθέτουμε το σύνολο του κώδικα της εφαρμογής μας. Αντί αυτού, επιλέγουμε να παραθέσουμε ενδεικτικά κομμάτια από τον κώδικα αυτό που θεωρούμε ότι ίσως είναι τα πιο σημαντικά. Ο κώδικας που παρατίθεται είναι βέβαια τεκμηριωμένος με επεξηγηματικά σχόλια όπου χρειάζεται.

Ακολουθεί ο κώδικας των αρχείων `VasInterfaceViewController.h` και `VasInterfaceViewController.m` που αντιστοιχούν στην αρχική οθόνη της εφαρμογής μας. Σημαντικό μέρος του κώδικα αυτών των αρχείων χρησιμοποιείται χωρίς μεγάλες διαφορές και στα υπόλοιπα μέρη της εφαρμογής που έχουν να κάνουν με το interface. Περιλαμβάνει κατά βάση συναρτήσεις που ορίζουν τις ενέργειες που πρέπει να γίνουν ως απόκριση σε ενέργειες του χρήστη (πάτημα κουμπιών, gestures κτλ.).

VasInterfaceViewController.h

```
//
//  VASInterfaceViewController.h
//  VASInterface
//
//  Created by zekesp on 11/3/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>
#import "VASBrain.h"
#import "GraphCalcViewController.h"

@interface VASInterfaceViewController : UIViewController <UITextFieldDelegate> {
        IBOutlet UIView *portrait;
        IBOutlet UIView *landscape;
        IBOutlet UITextField *polyTextfield;
        IBOutlet UITextField *precisionDigitsTextfield;
        IBOutlet UITextField *polyTextfieldLandscape;
        IBOutlet UITextField *precisionDigitsTextfieldLandscape;
        IBOutlet UIActivityIndicatorView *spinnerVAS;
        IBOutlet UIActivityIndicatorView *spinnerSturm;
        IBOutlet UIActivityIndicatorView *spinnerAppr;
        IBOutlet UIActivityIndicatorView *spinnerVASLandscape;
        IBOutlet UIActivityIndicatorView *spinnerSturmLandscape;
        IBOutlet UIActivityIndicatorView *spinnerApprLandscape;
        NSString *currentPolyInput;
        NSString *currentPrecisionDigitsInput;
        VASBrain *brain;
        GraphCalcViewController *graphCalcViewController;  //for the pinch
}

@property(nonatomic,retain) IBOutlet UIView *portrait;
@property(nonatomic,retain) IBOutlet UIView *landscape;
@property(nonatomic,retain) IBOutlet UITextField *polyTextfield;
@property(nonatomic,retain) IBOutlet UITextField *precisionDigitsTextfield;
@property(nonatomic,retain) IBOutlet UITextField *polyTextfieldLandscape;
@property(nonatomic,retain) IBOutlet UITextField *precisionDigitsTextfieldLandscape;
@property(nonatomic,retain) IBOutlet UIActivityIndicatorView *spinnerVAS;
@property(nonatomic,retain) IBOutlet UIActivityIndicatorView *spinnerSturm;
@property(nonatomic,retain) IBOutlet UIActivityIndicatorView *spinnerAppr;
@property(nonatomic,retain) IBOutlet UIActivityIndicatorView *spinnerVASLandscape;
@property(nonatomic,retain) IBOutlet UIActivityIndicatorView *spinnerSturmLandscape;
@property(nonatomic,retain) IBOutlet UIActivityIndicatorView *spinnerApprLandscape;

@property(nonatomic, retain) NSString *currentPolyInput;
@property(nonatomic, retain) NSString *currentPrecisionDigitsInput;
@property(retain) GraphCalcViewController *graphCalcViewController;

-(IBAction)infoPressed:(UIButton *)sender;
-(IBAction)VASPressed:(UIButton *)sender;
-(IBAction)SturmPressed:(UIButton *)sender;
-(IBAction)approximateRootsPressed:(UIButton *)sender;
-(IBAction)moreFunctionsPressed:(UIButton *)sender;
-(IBAction)graphPressed:(UIButton *)sender;

// for hiding the keyboard
-(IBAction)textFieldReturn:(id)sender;
-(IBAction)backgroundTouched:(id)sender;

@end
```

VasInterfaceViewController.m

```
//
//  VASInterfaceViewController.m
//  VASInterface
//
//  Created by zekesp on 11/3/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import "VASInterfaceViewController.h"
#import "IntervalsResultsViewController.h"
#import "RootsResultsViewController.h"
#import "MoreFunctionsViewController.h"
#import "GraphCalcViewController.h"

@implementation VASInterfaceViewController

@synthesize graphCalcViewController;
@synthesize portrait,landscape;
@synthesize polyTextfield, precisionDigitsTextfield;         // portrait textfields
@synthesize polyTextfieldLandscape, precisionDigitsTextfieldLandscape;   // landscape
@synthesize currentPolyInput, currentPrecisionDigitsInput;
@synthesize spinnerVAS,spinnerSturm,spinnerAppr;
@synthesize spinnerVASLandscape,spinnerSturmLandscape,spinnerApprLandscape;

- (id)init {
        if (self = [super init]) {
                GraphCalcViewController *gvc = [[GraphCalcViewController alloc] init];
                self.graphCalcViewController = gvc;
                [gvc release];

                self.graphCalcViewController.graphScale = 14;
        }
        return self;
}

- (void) showGeneralInfoView:(id)sender {
        UIAlertView *generalInfoAlert = [[UIAlertView alloc]
                                                initWithTitle:@"Polynomial Real Roots"
                                                message:@"This application was created by
Spyros Kehagias under the supervision of Alkiviadis G. Akritas as a thesis for the
Department of Computer & Communication Engineering of the University of Thessaly,
Greece.\n\nIt uses Giac, a free (GPL) C++ library created by Bernard Parisse."
                                                delegate:nil
                                                cancelButtonTitle:@"OK"
                                                otherButtonTitles:nil];
        [generalInfoAlert show];
        [generalInfoAlert release];
}

-(void)saveCurrentInput:(NSString *)currentInputPoly {
        // save last typed in polynomial (it will be restored next time we run the ap-
plication)
        NSUserDefaults *savedData = [NSUserDefaults standardUserDefaults];

        if(((self.interfaceOrientation == UIInterfaceOrientationPortrait) ||
                (self.interfaceOrientation==UIInterfaceOrientationPortraitUpsideDown)))
        {
                [savedData setObject:polyTextfield.text forKey:@"polyTextfield1"];
        }
        else if(((self.interfaceOrientation == UIInterfaceOrientationLandscapeLeft) ||
                (self.interfaceOrientation == UIInterfaceOrientationLandscapeRight))) {
```

```
                [savedData setObject:polyTextfieldLandscape.text
forKey:@"polyTextfield1"];
        }
        [[NSUserDefaults standardUserDefaults] synchronize];
}

-(void)restoreSavedInput {
        // restored last typed in polynomial

        if ([[NSUserDefaults standardUserDefaults] stringForKey:@"polyTextfield1"]){
                [polyTextfield setText:[[NSUserDefaults standardUserDefaults] string-
ForKey:@"polyTextfield1"]];
        }
        return;
}

- (void)viewDidLoad {
        [super viewDidLoad];
        self.title = [NSString stringWithFormat: @"Polynomial Real Roots"];

        // create model
        if(!brain) { brain = [[VASBrain alloc] init]; }

        // Navigation Bar info button
        UIButton* generalInfoButton = [UIButton buttonWithType:UIButtonTypeInfoLight];
        [generalInfoButton addTarget:self action:@selector(showGeneralInfoView:)
forControlEvents:UIControlEventTouchUpInside];
        self.navigationItem.rightBarButtonItem = [[UIBarButtonItem alloc] initWithCus-
tomView:generalInfoButton];
        //[[self navigationItem] setTitle:@"Some Title"];

        NSUserDefaults *prefs = [NSUserDefaults standardUserDefaults];
        self.graphCalcViewController.graphScale = [prefs floatForKey:@"graphScale"];
        [self.graphCalcViewController.graphView setNeedsDisplay];

        [self restoreSavedInput];
}

-(void)viewWillAppear:(BOOL)animated {
        // choose the right views for each orientation (if needed)

        if(((self.interfaceOrientation == UIInterfaceOrientationPortrait) ||
                (self.interfaceOrientation == UIInterfaceOrientationPortraitUp-
sideDown))) {
                if(self.view == self.landscape) { self.view = self.portrait; }
        }
        else if(((self.interfaceOrientation == UIInterfaceOrientationLandscapeLeft) ||
                        (self.interfaceOrientation == UIInterfaceOrientationLand-
scapeRight))) {
                if(self.view == self.portrait) { self.view = self.landscape; }
        }
}

- (IBAction)textFieldReturn:(id)sender {
        // hide keyboard when 'return' is pressed

        [sender resignFirstResponder];
        [self saveCurrentInput:polyTextfield.text];
}

-(IBAction)backgroundTouched:(id)sender {
        // hide keyboard when background is touched

        [precisionDigitsTextfield resignFirstResponder];
        [polyTextfield resignFirstResponder];
```

```
        [precisionDigitsTextfieldLandscape resignFirstResponder];
        [polyTextfieldLandscape resignFirstResponder];
        [self saveCurrentInput:polyTextfield.text];
}

-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientat
ion {
    // support all orientations (portrait & landscape)

        if(((interfaceOrientation == UIInterfaceOrientationLandscapeLeft) ||
                (interfaceOrientation == UIInterfaceOrientationLandscapeRight))) {
                self.view = landscape;
        }
        else if (((interfaceOrientation == UIInterfaceOrientationPortrait) ||
                        (interfaceOrientation == UIInterfaceOrientationPortraitUp-
sideDown))) {
                self.view = portrait;
        }
        return YES;
}

-(void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation
duration:(NSTimeInterval)duration {
        // temporarily save current poly & precision digits, so they can be retrieved
after rotation

        if(((self.interfaceOrientation == UIInterfaceOrientationPortrait) ||
                (self.interfaceOrientation == UIInterfaceOrientationPortraitUp-
sideDown))) {
                self.currentPolyInput = self.polyTextfield.text;
                self.currentPrecisionDigitsInput = self.precisionDigitsTextfield.text;
        }
        else if(((self.interfaceOrientation == UIInterfaceOrientationLandscapeLeft) ||
                     (self.interfaceOrientation == UIInterfaceOrientationLand-
scapeRight))) {
                self.currentPolyInput = self.polyTextfieldLandscape.text;
                self.currentPrecisionDigitsInput =
self.precisionDigitsTextfieldLandscape.text;
        }
}

-
(void)didRotateFromInterfaceOrientation:(UIInterfaceOrientation)fromInterfaceOrientati
on {
        // retrieve previously saved poly & precision digits after rotation

        if(((self.interfaceOrientation == UIInterfaceOrientationPortrait) ||
                (self.interfaceOrientation == UIInterfaceOrientationPortraitUp-
sideDown))) {
                self.polyTextfield.text = self.currentPolyInput;
                self.precisionDigitsTextfield.text = self.currentPrecisionDigitsInput;
        }
        else if(((self.interfaceOrientation == UIInterfaceOrientationLandscapeLeft) ||
                        (self.interfaceOrientation == UIInterfaceOrientationLand-
scapeRight))) {
                self.polyTextfieldLandscape.text = self.currentPolyInput;
                self.precisionDigitsTextfieldLandscape.text =
self.currentPrecisionDigitsInput;
        }
}

-(BOOL)polyTextfieldIsEmpty {
        // diplay alert window if polyTextfield is empty
```

```
        if(self.interfaceOrientation==UIInterfaceOrientationPortrait ||
           self.interfaceOrientation==UIInterfaceOrientationPortraitUpsideDown) {
                if([polyTextfield.text length]==0) {
                        UIAlertView *noPolyAlert = [[UIAlertView alloc]
                                                initWithTitle:@"Invalid Input"
                                                message:@"No polynomial inserted. Please
insert polynomial."
                                                delegate:nil
                                                cancelButtonTitle:@"OK"
                                                otherButtonTitles:nil];
                        [noPolyAlert show];
                        [noPolyAlert release];
                        return YES;
                }
        }

        else if(self.interfaceOrientation==UIInterfaceOrientationLandscapeLeft ||
                        self.interfaceOrientation==UIInterfaceOrientationLandscapeRight)
{
                if([polyTextfieldLandscape.text length]==0) {
                        UIAlertView *noPolyAlert = [[UIAlertView alloc]
                                                initWithTitle:@"Invalid Input"
                                                message:@"No polynomial inserted. Please
insert polynomial."
                                                delegate:nil
                                                cancelButtonTitle:@"OK"
                                                otherButtonTitles:nil];
                        [noPolyAlert show];
                        [noPolyAlert release];
                        return YES;
                }
        }
        return NO;
}

-(BOOL)precisionDigitsTextfieldIsEmpty {
        // diplay alert window if precisionDigitsTextfield is empty

        if(self.interfaceOrientation==UIInterfaceOrientationPortrait ||
           self.interfaceOrientation==UIInterfaceOrientationPortraitUpsideDown) {
                if([precisionDigitsTextfield.text length]==0) {
                        UIAlertView *noPolyAlert = [[UIAlertView alloc]
                                                initWithTitle:@"Invalid Input"
                                                message:@"No precision digits inserted.
Please insert precision digits."

                                                delegate:nil
                                                cancelButtonTitle:@"OK"
                                                otherButtonTitles:nil];
                        [noPolyAlert show];
                        [noPolyAlert release];
                        return YES;
                }
        }
        else if(self.interfaceOrientation==UIInterfaceOrientationLandscapeLeft ||
                        self.interfaceOrientation==UIInterfaceOrientationLandscapeRight)
{
                if([precisionDigitsTextfieldLandscape.text length]==0) {
                        UIAlertView *noPolyAlert = [[UIAlertView alloc]
                                                initWithTitle:@"Invalid Input"
                                                message:@"No precision digits inserted.
Please insert precision digits."
                                                delegate:nil
                                                cancelButtonTitle:@"OK"
                                                otherButtonTitles:nil];
```

```
                        [noPolyAlert show];
                        [noPolyAlert release];
                        return YES;
                }
        }
        return NO;
}

-(BOOL)precisionDigitsInputInvalid {
        // diplay alert window if precisionDigitsTextfield has invalid input

        if(self.interfaceOrientation==UIInterfaceOrientationPortrait ||
           self.interfaceOrientation==UIInterfaceOrientationPortraitUpsideDown) {
                if([precisionDigitsTextfield.text intValue]<0 || [precisionDigitsText-
field.text intValue]>11) {
                        UIAlertView *noPolyAlert = [[UIAlertView alloc]
                                                initWithTitle:@"Invalid Input"
                                                message:@"Invalid input. Please insert
correct number of precision digits (between 0 and 11)."
                                                delegate:nil
                                                cancelButtonTitle:@"OK"
                                                otherButtonTitles:nil];
                        [noPolyAlert show];
                        [noPolyAlert release];
                        return YES;
                }
        }

        else if(self.interfaceOrientation==UIInterfaceOrientationLandscapeLeft ||
                        self.interfaceOrientation==UIInterfaceOrientationLandscapeRight)
{
                if([precisionDigitsTextfieldLandscape.text intValue]<0 ||
                   [precisionDigitsTextfieldLandscape.text intValue]>11) {
                        UIAlertView *noPolyAlert = [[UIAlertView alloc]
                                                initWithTitle:@"Invalid Input"
                                                message:@"Invalid Input. Please insert
correct number of precision digits (between 0 and 11)."
                                                delegate:nil
                                                cancelButtonTitle:@"OK"
                                                otherButtonTitles:nil];
                        [noPolyAlert show];
                        [noPolyAlert release];
                        return YES;
                }
        }
        return NO;
}

-(IBAction)infoPressed:(UIButton *)sender {
        // display info alert windows

        int tagNum = sender.tag;

        if(tagNum==1) {
                UIAlertView *instructionsAlert = [[UIAlertView alloc]
                                        initWithTitle:@"Instructions"
                                        message:@"Insert a polynomial into the text-
field. Example given: 5x-3x^3+6\n(No need to use the * sign for multiplica-
tion).\n\nVAS Real Intervals will compute the real root isolation intervals of the
poly using Vincent's theorem of 1836 whereas Sturm Real Intervals does the same using
Sturm's theorem of 1827. Details can be found in the web."
                                        delegate:nil
                                        cancelButtonTitle:@"OK"
                                        otherButtonTitles:nil];
                [instructionsAlert show];
```

```
                [instructionsAlert release];
        }
        if(tagNum==2) {
                UIAlertView *vasInfoAlert = [[UIAlertView alloc]
                                        initWithTitle:@"VAS Real Roots"
                                        message:@"Uses the Vincent-Akritas-Strzebonski
method for the isolation of the real roots of polynomials with integer coefficients.
\n\nReturns a list of the isolation intervals of the roots along with their multiplic-
ity."
                                        delegate:nil
                                        cancelButtonTitle:@"OK"
                                        otherButtonTitles:nil];
                [vasInfoAlert show];
                [vasInfoAlert release];
        }

        if(tagNum==3) {
                UIAlertView *sturmInfoAlert = [[UIAlertView alloc]
                                         initWithTitle:@"Sturm Real Roots"
                                         message:@"Uses Sturm's method for the isolation
of the real roots of polynomials with integer coefficients. \n\nReturns a list of the
isolation intervals of the roots along with their multiplicity."
                                         delegate:nil
                                        cancelButtonTitle:@"OK"
                                        otherButtonTitles:nil];
                [sturmInfoAlert show];
                [sturmInfoAlert release];
        }

        if(tagNum==4) {
                UIAlertView *approxInfoAlert = [[UIAlertView alloc]
                                        initWithTitle:@"Approximate Real Roots"
                                        message:@"Approximates the roots in the isolating
intervals computed by 'VAS Real Roots'. Up to 11 precision digits allowed."
                                        delegate:nil
                                        cancelButtonTitle:@"OK"
                                        otherButtonTitles:nil];
                [approxInfoAlert show];
                [approxInfoAlert release];
        }

        if(tagNum==5) {
                UIAlertView *graphInfoAlert = [[UIAlertView alloc]
                                        initWithTitle:@"Graph"
                                        message:@"Plots a graph of the polynomial func-
tion defined by the given polynomial.\n\nGestures allowed:\n\nPanning: drags the graph
around - with one finger\n\nPinching: zooms in and out\n\nTapping twice: brings the
axis to the center of the screen"
                                        delegate:nil
                                        cancelButtonTitle:@"OK"
                                        otherButtonTitles:nil];
                [graphInfoAlert show];
                [graphInfoAlert release];
        }
}

-(void)showInvalidInputError {
        UIAlertView *invalidInputAlert = [[UIAlertView alloc]
                                         initWithTitle:@"ERROR!"
                                         message:@"Invalid input. Please insert correct
polynomial.\n\n(Correct polynomials have only one variable, preferrably 'x' and no
special characters like '#','$','&','@','!' etc.)"
                                         delegate:nil
                                         cancelButtonTitle:@"OK"
                                         otherButtonTitles:nil];
```

```
	[invalidInputAlert show];
	[invalidInputAlert release];
}

-(IBAction)VASPressed:(UIButton *)sender
{
	// when 'VAS' button is pressed, call corresponding 'brain' method to compute
the isolation intervals

	BOOL validInput;

	if([self polyTextfieldIsEmpty]) { return; }
	else {
		if(self.interfaceOrientation==UIInterfaceOrientationPortrait ||
		   self.interfaceOrientation==UIInterfaceOrientationPortraitUpsideDown)
{
			[NSThread detachNewThreadSelector:@selector(startAnimating) to-
Target:spinnerVAS withObject:nil];
			validInput = [brain getIntervals:polyTextfield.text with:@"VAS"
andPrecisionDigits:6];
			[NSThread detachNewThreadSelector:@selector(stopAnimating) to-
Target:spinnerVAS withObject:nil];
		}
		else if(self.interfaceOrientation==UIInterfaceOrientationLandscapeLeft
|| self.interfaceOrientation==UIInterfaceOrientationLandscapeRight) {
			[NSThread detachNewThreadSelector:@selector(startAnimating) to-
Target:spinnerVASLandscape withObject:nil];
			validInput = [brain getIntervals:polyTextfieldLandscape.text
with:@"VAS" andPrecisionDigits:6];
			[NSThread detachNewThreadSelector:@selector(stopAnimating) to-
Target:spinnerVASLandscape withObject:nil];
		}
		if(!validInput) {
			[self showInvalidInputError];
			return;
		}
	}

	// push the ViewController responsible for showing the VAS isolation intervals
on the stack

	IntervalsResultsViewController *vasResultsVC = [[IntervalsResultsViewController
alloc] init];
	vasResultsVC.intervalsString = brain.intervals;
	vasResultsVC.poly = brain.polynomial;
	vasResultsVC.compTime = brain.computationTime;
	vasResultsVC.whichMethod = @"VAS";
	vasResultsVC.title = [NSString stringWithFormat:@"VAS Real Intervals"];
	[self.navigationController pushViewController:vasResultsVC animated:YES];
	[vasResultsVC release];
}

-(IBAction)SturmPressed:(UIButton *)sender
{
	// when 'Sturm' button is pressed, call corresponding 'brain' method to compute
the isolation intervals

	BOOL validInput;

	if([self polyTextfieldIsEmpty]) { return; }
	else {
		if([self precisionDigitsTextfieldIsEmpty] || [self precisionDig-
itsInputInvalid])
			{ return; }
		else {
```

```
                    if(self.interfaceOrientation==UIInterfaceOrientationPortrait ||
            self.interfaceOrientation==UIInterfaceOrientationPortraitUpsideDown) {
                            [NSThread detachNewThreadSelec-
tor:@selector(startAnimating) toTarget:spinnerSturm withObject:nil];
                            validInput = [brain getIntervals:polyTextfield.text
with:@"Sturm" andPrecisionDigits:[precisionDigitsTextfield.text intValue]];

                            [NSThread detachNewThreadSelec-
tor:@selector(stopAnimating) toTarget:spinnerSturm withObject:nil];
                    }
                    else
if(self.interfaceOrientation==UIInterfaceOrientationLandscapeLeft ||

       self.interfaceOrientation==UIInterfaceOrientationLandscapeRight) {
                            [NSThread detachNewThreadSelec-
tor:@selector(startAnimating) toTarget:spinnerSturmLandscape withObject:nil];
                            validInput = [brain getInter-
vals:polyTextfieldLandscape.text with:@"Sturm" andPrecisionDig-
its:[precisionDigitsTextfieldLandscape.text intValue]];
                            [NSThread detachNewThreadSelec-
tor:@selector(stopAnimating) toTarget:spinnerSturmLandscape withObject:nil];

                    }
            }
            if(!validInput) {
                    [self showInvalidInputError];
                    return;
            }
       }

       // push the ViewController responsible for showing the Sturm isolation inter-
vals on the stack

       IntervalsResultsViewController *sturmResultsVC = [[IntervalsResultsViewControl-
ler alloc] init];
       sturmResultsVC.intervalsString = brain.intervals;
       sturmResultsVC.poly = brain.polynomial;
       sturmResultsVC.compTime = brain.computationTime;
       sturmResultsVC.whichMethod = @"Sturm";
       sturmResultsVC.title = [NSString stringWithFormat:@"Sturm Real Intervals"];
       [self.navigationController pushViewController:sturmResultsVC animated:YES];
       [sturmResultsVC release];
}

-(IBAction)approximateRootsPressed:(UIButton *)sender
{
       // when 'approximate' button is pressed, call corresponding 'brain' method to
compute the
       // real roots based on the isolation intervals retrieved by the 'VAS' 'brain'
function

       BOOL validInput;

       if([self polyTextfieldIsEmpty]) { return; }
       else {
            if([self precisionDigitsTextfieldIsEmpty] || [self precisionDig-
itsInputInvalid])
                    { return; }
            else {
                    if(self.interfaceOrientation==UIInterfaceOrientationPortrait ||

self.interfaceOrientation==UIInterfaceOrientationPortraitUpsideDown) {
                            [NSThread detachNewThreadSelec-
tor:@selector(startAnimating) toTarget:spinnerAppr withObject:nil];
```

```
                                validInput = [brain getRoots:polyTextfield.text withPre-
cisionDigits:[precisionDigitsTextfield.text intValue]];
                                [NSThread detachNewThreadSelec-
tor:@selector(stopAnimating) toTarget:spinnerAppr withObject:nil];

                        }
                        else
if(self.interfaceOrientation==UIInterfaceOrientationLandscapeLeft ||

       self.interfaceOrientation==UIInterfaceOrientationLandscapeRight) {
                                [NSThread detachNewThreadSelec-
tor:@selector(startAnimating) toTarget:spinnerApprLandscape withObject:nil];
                                validInput = [brain getRoots:polyTextfieldLandscape.text
withPrecisionDigits:[precisionDigitsTextfieldLandscape.text intValue]];
                                [NSThread detachNewThreadSelec-
tor:@selector(stopAnimating) toTarget:spinnerApprLandscape withObject:nil];

                        }
                }
                if(!validInput) {
                        [self showInvalidInputError];
                        return;
                }
        }

        // push the ViewController responsible for showing the approximated real roots
on the stack

        RootsResultsViewController *rootsResultsVC = [[RootsResultsViewController al-
loc] init];
        rootsResultsVC.rootsString = brain.rootList;
        rootsResultsVC.poly = brain.polynomial;
        rootsResultsVC.whichRoots = @"approximated";
        rootsResultsVC.title = [NSString stringWithFormat:@"Approx. VAS Roots"];
        [self.navigationController pushViewController:rootsResultsVC animated:YES];
        [rootsResultsVC release];
}

-(void)graphFunction:(NSString *)function
{
        if (self.graphCalcViewController.graphScale == 0) {
                self.graphCalcViewController.graphScale = 14;
        }

        if(self.interfaceOrientation==UIInterfaceOrientationPortrait ||
           self.interfaceOrientation==UIInterfaceOrientationPortraitUpsideDown) {
                self.graphCalcViewController.functionToGraph = [VASBrain getNormal-
Poly:polyTextfield.text];
                self.graphCalcViewController.title = polyTextfield.text;

        }
        else if(self.interfaceOrientation==UIInterfaceOrientationLandscapeLeft ||
                        self.interfaceOrientation==UIInterfaceOrientationLandscapeRight)
{
                self.graphCalcViewController.functionToGraph =
                                [VASBrain getNormalPoly:polyTextfieldLandscape.text];
                self.graphCalcViewController.title = polyTextfieldLandscape.text;
        }

        // setting the scale with the buttons
        NSUserDefaults *prefs = [NSUserDefaults standardUserDefaults];
        [prefs setFloat: self.graphCalcViewController.graphScale forKey:@"graphScale"];
        [[NSUserDefaults standardUserDefaults] synchronize];
```

```
        [self.graphCalcViewController.graphView setNeedsDisplay]; // added for the
pinch to stay

        if (self.graphCalcViewController.view.window == nil) {
                [self.navigationController pushViewControl-
ler:self.graphCalcViewController animated:YES];
        }
}

-(IBAction)graphPressed:(UIButton *)sender
{
        if(self.interfaceOrientation==UIInterfaceOrientationPortrait ||
           self.interfaceOrientation==UIInterfaceOrientationPortraitUpsideDown) {
                if (![self polyTextfieldIsEmpty]) {

                        // check if inserted poly is valid
                        if([[VASBrain getNormalPoly:self.polyTextfield.text] isE-
qual:@"not a poly"]) {
                                [self showInvalidInputError];
                                return;
                        }
                        [self graphFunction: polyTextfield.text];
                }

        }
        else if(self.interfaceOrientation==UIInterfaceOrientationLandscapeLeft ||
                        self.interfaceOrientation==UIInterfaceOrientationLandscapeRight)
{
                if (![self polyTextfieldIsEmpty]) {

                        // check if inserted poly is valid
                        if([[VASBrain getNormalPoly:self.polyTextfieldLandscape.text]
isEqual:@"not a poly"]) {
                                [self showInvalidInputError];
                                return;
                        }
                        [self graphFunction: polyTextfieldLandscape.text];
                }
        }
}

-(NSString *)retrievePoly
{
        // retrieve the poly typed in the polyTextfield so it can be shown on the 'More
functions' screen as well

        if(self.interfaceOrientation==UIInterfaceOrientationPortrait ||
           self.interfaceOrientation==UIInterfaceOrientationPortraitUpsideDown) {
                return self.polyTextfield.text;
        }
        else if(self.interfaceOrientation==UIInterfaceOrientationLandscapeLeft ||
                        self.interfaceOrientation==UIInterfaceOrientationLandscapeRight)
{
                return self.polyTextfieldLandscape.text;
        }
        return @"";
}

-(IBAction)moreFunctionsPressed:(UIButton *)sender
{
        // push the ViewController responsible for showing the 'More functions' screen
on the stack

        MoreFunctionsViewController *moreFunctionsVC = [[MoreFunctionsViewController
alloc] init];
```

```
        moreFunctionsVC.title = [NSString stringWithFormat:@"More Functions"];
        moreFunctionsVC.currentPolyInput = [self retrievePoly];
        [self.navigationController pushViewController:moreFunctionsVC animated:YES];
        [moreFunctionsVC release];
}

- (void)didReceiveMemoryWarning {
        // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];
        // Release any cached data, images, etc that aren't in use.
}

-(void) releaseOutlets
{
        self.landscape = nil;
        self.portrait = nil;
        self.polyTextfield = nil;
        self.precisionDigitsTextfield = nil;
        self.polyTextfieldLandscape = nil;
        self.precisionDigitsTextfieldLandscape = nil;
        self.spinnerVAS = nil;
        self.spinnerSturm = nil;
        self.spinnerAppr = nil;
        self.spinnerVASLandscape = nil;
        self.spinnerApprLandscape = nil;
        self.spinnerSturmLandscape = nil;
        self.navigationItem.rightBarButtonItem = nil;
}

- (void)viewDidUnload {
        // Release any retained subviews of the main view.
        // e.g. self.myOutlet = nil;
        [super viewDidUnload];
        [self releaseOutlets];
}

- (void)dealloc {
        [brain release];
        [self releaseOutlets];
    [super dealloc];
}

@end
```

Στη συνέχεια παραθέτουμε τον κώδικα των αρχείων VASBrain.h και VASBrain.mm, που αποτελούν το μαθηματικό «εγκέφαλο» της εφαρμογής μας (εδώ χρησιμοποιούνται οι συναρτήσεις της Giac). Περιέχουν τις συναρτήσεις που υλοποιούν τον αλγόριθμο VAS αλλά και τις άλλες λειτουργίες που προσφέρει η εφαρμογή Polynomial Real Roots.

VASBrain.h

```
//
//  VASBrain.h
//  VASInterface3(withNavCon)
//
//  Created by zekesp on 11/6/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import <Foundation/Foundation.h>

@interface VASBrain : NSObject {
@private
        NSString *intervals;
        NSString *polynomial;
        NSString *computationTime;
        NSString *rootList;
        NSString *upperBoundStr;
        NSString *lowerBoundStr;
        NSString *factoredPolyStr;
        NSString *factoredSqrFreePolyStr;
}

@property (nonatomic, retain) NSString *intervals;
@property (nonatomic, retain) NSString *polynomial;
@property (nonatomic, retain) NSString *computationTime;
@property (nonatomic, retain) NSString *rootList;
@property (nonatomic, retain) NSString *upperBoundStr;
@property (nonatomic, retain) NSString *lowerBoundStr;
@property (nonatomic, retain) NSString *factoredPolyStr;
@property (nonatomic, retain) NSString *factoredSqrFreePolyStr;

- (BOOL)getIntervals:(NSString *)inputPoly with:(NSString *)method andPrecisionDig-
its:(int)precDigits;
- (BOOL)getRoots:(NSString *)inputPoly withPrecisionDigits:(int)precDigits;
- (BOOL)getBound:(NSString *)inputPoly kind:(NSString *)kindOfBound;
- (BOOL)getFactoredPoly:(NSString *)inputPoly kind:(NSString *)kindOfFactorization;
- (BOOL)getRationalRoots:(NSString *)inputPoly;
+ (NSString *)getNormalPoly:(NSString *)inputPoly;

@end
```

VASBrain.mm

```
//
//  VASBrain.mm
//  VASInterface
//
//  Created by zekesp on 11/6/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import "VASBrain.h"

//#include "gmp.h"
//#include "gmpxx.h"
#include <giac/giac.h>
#include <giac/misc.h>
#include <giac/csturm.h>
#include <string>
#include <iostream>
#include <sys/time.h>
#include <stdlib.h>

using namespace std;
using namespace giac;

@implementation VASBrain

@synthesize intervals,polynomial,computationTime,rootList;
@synthesize upperBoundStr,lowerBoundStr,factoredPolyStr,factoredSqrFreePolyStr;

static struct timeval start_time1;
typedef long long int64;
void init_time() {
	gettimeofday(&start_time1, NULL);
}
int64 get_time1() {
	struct timeval t;
	gettimeofday(&t, NULL);
	return (int64) (t.tv_sec - start_time1.tv_sec) * 1000000 + (t.tv_usec -
start_time1.tv_usec);
}

void printGenType(gen e){
	if (e.type==_INT_)		{ NSLog(@"gen type: int (0)"); }
	if (e.type==_DOUBLE_)	{ NSLog(@"gen type: double (1)"); }
	if (e.type==_ZINT)		{ NSLog(@"gen type: mpz_t (2)"); }
	if (e.type==_REAL)		{ NSLog(@"gen type: mpf_t (3)"); }
	if (e.type==_CPLX)		{ NSLog(@"gen type: gen (4)"); }
	if (e.type==_POLY)		{ NSLog(@"gen type: polynome (5)"); }
	if (e.type==_IDNT)		{ NSLog(@"gen type: identificateur (6)"); }
	if (e.type==_VECT)		{ NSLog(@"gen type: vecteur (7)"); }
	if (e.type==_SYMB)		{ NSLog(@"gen type: symbolic (8)"); }
	if (e.type==_SPOL1)		{ NSLog(@"gen type: sparse_poly1 (9)"); }
	if (e.type==_FRAC)		{ NSLog(@"gen type: fraction (10)"); }
	if (e.type==_EXT)		{ NSLog(@"gen type: gen (11)"); }
	if (e.type==_STRNG)		{ NSLog(@"gen type: string (12)"); }
	if (e.type==_FUNC)		{ NSLog(@"gen type: unary_fonction_ptr (13)"); }
	if (e.type==_ROOT)		{ NSLog(@"gen type: real_complex_rootof (14)"); }
	if (e.type==_MOD)		{ NSLog(@"gen type: gen (15)"); }
	if (e.type==_USER)		{ NSLog(@"gen type: gen_user (16)"); }
	if (e.type==_MAP)		{ NSLog(@"gen type: map<gen.gen> (17)"); }
	if (e.type==_EQW)		{ NSLog(@"gen type: eqwdata (18)"); }
	if (e.type==_GROB)		{ NSLog(@"gen type: grob (19)"); }
```

```
        if (e.type==_POINTER_)    { NSLog(@"gen type: void (20)"); }
        if (e.type==_FLOAT_)      { NSLog(@"gen type: immediate (21)"); }
}
void printSymbGen(gen e){
    string str1 = print(e,0);      // print() returns a string of 'gen'
    const char *str = str1.c_str();// convert 'string' to C string (char *)
    NSString *s = [[NSString alloc] initWithUTF8String:str];  // convert C string to
NSString
    NSLog(@"result: %@",s);           // print NSString on the console
    [s release];
}
void pg(gen e) {
        printGenType(e);
        printSymbGen(e);
}
int isPoly(gen e) {
        e=_symb2poly(e,0);
        vecteur &v1 = *e._VECTptr;
        if(print(v1,0).find("undef")!=-1) { return -1; }
        else  { return 1; }
}
//gen toReal(gen e){
//      return gen(real_object(e));
//}
gen my_sqrfree(gen p){
        //---gets from sqrfree(p) the list of polys with simple roots
        //---along with their multiplicities

        gen a, ekthetes;

        //---remove the content for smaller coefficients (and for sqrfree)
        p = giac::normal(p/_gcd(_symb2poly(p,0),0),0);

        a = _sqrfree(p,0);

        //---poly is sqrfree
        if (a == p) { return makevecteur(makevecteur(a,1)); }

        //---poly has multiple roots
        ekthetes = _newList(_size(a,0),0);
        vecteur &v1 = *ekthetes._VECTptr;

        //---only one factor raised to a power > 1!
        //---(one factor raised to power 1 means poly is sqrfree!)

        // a[0]._FUNCptr->ptr()->s returns '^'
        if((a[0]._FUNCptr->ptr()->s)[0] == '^') { return makevec-
teur(makevecteur(a[1],a[2])); }

        //---more than one factors raised to powers >= 1
        if ((a[0]._FUNCptr->ptr()->s)[0] == '*') {
                for(int k=1; k<=_size(a,0).val; k++) {

                        const char *tempStr = print(a[k][0],0).c_str();

                        //---factor raised to power == length(a[k]) > 1
                        if(tempStr[1] == '^') {       v1[k-1] = makevec-
teur(a[k][1],(a[k][_size(a[k],0).val])); }
                        //---factor raised to power == 1
                        if(tempStr[1] == '+') {       v1[k-1] = makevecteur(a[k],1); }
                }
        }
        return(ekthetes);
}
gen my_sqrfree2(gen p){
```

```
        //---gets from sqrfree(p) the list of polys with simple roots
        //---and multiplies them out.

        gen a, factorsProduct;

        //---remove the content for smaller coefficients (and for sqrfree)

        p = giac::normal(p/_gcd(_symb2poly(p,0),0),0);

        a = _sqrfree(p,0);

        const char *tempStr = print(a[0],0).c_str();

        //---poly is sqrfree with simple roots
        if (a == p && tempStr[1]=='+') { return a; }

        //---poly has multiple roots

        //---only one factor raised to a power > 1!
        //---(one factor raised to power 1 means poly is sqrfree!)
        if(tempStr[1]=='^') { return a[1]; }

        factorsProduct= gen(1,0);

        //---more than one factors raised to powers >= 1
        if (tempStr[1] == '*') {
                for(int k=1; k<=_size(a,0).val; k++) {

                        const char *tempStr1 = print(a[k][0],0).c_str();

                        if(tempStr1[1]=='^'){     //---factor raised to power ==
length(a[k]) > 1
                                factorsProduct = factorsProduct*a[k][1];
                        }
                        if(tempStr1[1]=='+'){     //---factor raised to power == 1
                                factorsProduct = factorsProduct*a[k];
                        }
                }
        }
        return(factorsProduct);
}
gen bisectionMethod_R(gen f, gen x, gen l, gen r, gen eps, gen maxIter){
        //---To be used ONLY when we have removed the rational roots first!!!
        //---That is, f has no rational roots!
        gen ll,rr,mi,sll,srr,smi;
        int j1;

        ll = l;
        rr = r;

        if (is_strictly_greater(ll,rr,0)) {
                NSLog(@"No proper interval");
                return 0;
        }

        // form expressions "x=ll","x=rr"
        string s1 = "x=" + print(ll,0);
        string s2 = "x=" + print(rr,0);
        string s3;

        sll = _subst(makevecteur(f,gen(s1,0)),0);
        srr = _subst(makevecteur(f,gen(s2,0)),0);

        if (is_strictly_greater(sll*srr,0,0)) {
                NSLog(@"No proper interval");
```

```
	}

	for(j1=1; j1<=maxIter.val; j1++) {
		mi = (ll+rr)/2;
		// form expression "x=mi"
		s3 = "x=" + print(mi,0);
		smi = _subst(makevecteur(f,gen(s3,0)),0);

		if (smi==0) { return mi; }	//---found the root
		if (is_strictly_greater(sll*smi,0,0)) {//---same signs; update interval
			ll = mi;
			sll = smi;
		}
		else {	//---opposite signs; update interval
			rr = mi;
			srr = smi;
		}
		if (is_greater(eps,_abs(rr-ll,0),0)) {
			return _evalf(mi,0);
			//return toReal(mi);
		}
	}
	return _evalf(mi,0);
	//return toReal(mi);
}
gen combineSublistsOfSameMultiplicity(gen intervalsLIST){
	//---merges sublists of the form [[[[0,0]],1],[[[-1/2,-1/2],[-
2,0]],1],[[[1/2,1/2],[0,2]],1],...] to
	//---[[[[0,0],[-1/2,-1/2],[1/2,1/2],[-2,0],[0,2]],1],...]

	gen rootIntervalsSameMultiplicity, temp, intervalsLISTP, intervalsLISTE;

	rootIntervalsSameMultiplicity = gen("[]",0);
	intervalsLISTP = _convert(makevecteur(intervalsLIST,gen("list",0)),0);

	intervalsLISTE = _head(intervalsLISTP,0);
	intervalsLISTP = _tail(intervalsLISTP,0);

	while (_size(intervalsLISTE,0).val!=0 && _size(intervalsLISTP,0).val!=0) {
		if (_head(_tail(intervalsLISTE,0),0) ==
_head(_tail(_head(intervalsLISTP,0),0),0)) {
			temp =
_union(makevecteur(_head(intervalsLISTE,0),_head(_head(intervalsLISTP,0),0)),0);
			rootIntervalsSameMultiplicity =
_append(makevecteur(rootIntervalsSameMultiplicity,

                                    makevec-
teur(temp,_head(_tail(intervalsLISTE,0),0))),0);
			intervalsLISTP = _tail(intervalsLISTP,0);
			intervalsLISTE = _head(intervalsLISTP,0);
			intervalsLISTP = _tail(intervalsLISTP,0);
		}
		else {
			rootIntervalsSameMultiplicity =

	_append(makevecteur(rootIntervalsSameMultiplicity,intervalsLISTE),0);
			intervalsLISTE = _head(intervalsLISTP,0);
			intervalsLISTP = _tail(intervalsLISTP,0);
		}
	}

	if (_size(intervalsLISTE,0).val!=0) {
		rootIntervalsSameMultiplicity =
	_append(makevecteur(rootIntervalsSameMultiplicity,intervalsLISTE),0);
	}
```

```
        return rootIntervalsSameMultiplicity;
}
gen formRationalRootIntervals(gen p){
        //---compute the rational roots and form their intervals with multiplicity;
        //---output a list of the form [[[a,a], m], ...]

        gen list_sqrfree_factors_plus_multiplicity;
        gen list_Of_RationalRoots_PlusMultiplicity_PerFactor, roots2intervals;
        gen sorted_SET;

        list_sqrfree_factors_plus_multiplicity = my_sqrfree(p);

        list_Of_RationalRoots_PlusMultiplicity_PerFactor =
        _map(makevecteur(list_sqrfree_factors_plus_multiplicity,gen("x-
>[rationalroot(x[0]),x[1]]",0)),0);

        roots2intervals =
_map(makevecteur(list_Of_RationalRoots_PlusMultiplicity_PerFactor,
gen("y->if (length(head(y)!=[])){[map(head(y),x->[x,x]),head(tail(y))]}",0)),0);

        return(roots2intervals);
}
gen addMissingSqrfreeFactors(gen pol, gen allIntervals){
        //---make sure list allIntervals has the same length as list pol; pol
        //---is a list of the form [[sqrfree_factor, m], ...]
        //---REASON: a rational root [[a,a], m] does NOT have the corresponding factor
in pol,
        //---so we insert in it the factor [x-a, m].

        gen pol2, polP, polE, allIntervalsP, allIntervalsE;

        pol2 = gen("[]",0);
        polP = pol;
        allIntervalsP = allIntervals; //---advance pointers

        polE = _head(polP,0);
        polP = _tail(polP,0);
        allIntervalsE = _head(allIntervalsP,0);
        allIntervalsP = _tail(allIntervalsP,0);

        while(_size(polE,0).val!=0 && _size(allIntervalsE,0).val!=0)
        {
                if (_head(_tail(allIntervalsE,0),0)==_head(_tail(polE,0),0)) {
                        pol2 = _append(makevecteur(pol2,polE),0);
                        polE = _head(polP,0);
                        polP = _tail(polP,0);
                        allIntervalsE = _head(allIntervalsP,0);
                        allIntervalsP = _tail(allIntervalsP,0);
                }
                else {
                        // form expression "x-head(head(allIntervalsE))[0]"
                        string s = "x-" + print(_head(_head(allIntervalsE,0),0)[0],0);

                        pol2 =
_append(makevecteur(pol2,makevecteur(gen(s,0),_head(_tail(allIntervals,0),0))),0);
                        allIntervalsE = _head(allIntervalsP,0);
                        allIntervalsP = _tail(allIntervalsP,0);
                }
        }
        return pol2;
}
gen interval_split(gen p1, gen a1){
        //---shortens an interval depending on where the root is

        gen a,b;
```

```
        a = a1;
        vecteur &v1 = *a._VECTptr;

        if(is_strictly_greater(gen(0,0),_horner(makevecteur(p1,a[0]),0)*

_horner(makevecteur(p1,((a[0]+a[1])/2)),0),0)) {     v1[1] = (a[0]+a[1])/2; }
        if(_horner(makevecteur(p1,(a[0]+a[1])/2),0)==0) {
                v1[0] = (a[0]+a[1])/2;
                v1[1] = (a[0]+a[1])/2;
        }

        if(is_strictly_greater(gen(0,0),_horner(makevecteur(p1,(a[0]+a[1])/2),0)*
                                                _horner(makevecteur(p1,a[1]),0),0)) {
v1[0] = (a[0]+a[1])/2; }
        return a;
}
gen interval_overlap2(gen pol, gen apot){
        //---a modification of the original interval_overlap to handle
        //---overlaps within intervals of roots of same multiplicity
        int kk,ll,pp;

        //---Cases intervals overlap: Let p1 have a root in ]a, b[
        //---                        and p2 have a root in ]c, d[.

        //---The intervals ]a, b[ and ]c, d[ overlap if::

        //---1. a<=c &&  b<=d && b>c
        //---OR
        //---2. a>=c && b>=d && d>a
        //---OR
        //---3. a<c && b>d
        //---OR
        //---4. a>c && b<d

        vecteur &v1 = *apot._VECTptr;

        for(kk=0; kk<_size(apot,0).val; kk++) {    //---for each list of the form
[[[a,a],[a,b]...],m]
                for(ll=0; ll<_size(apot[kk][0],0).val; ll++) {   //---for each list of
the form [[a,a],[a,b]...]
                        for(pp=ll+1; pp<_size(apot[kk][0],0).val; pp++) {

        while((is_greater(apot[kk][0][pp][0],apot[kk][0][ll][0],0) &&

is_greater(apot[kk][0][pp][1],apot[kk][0][ll][1],0) &&

is_strictly_greater(apot[kk][0][ll][1],apot[kk][0][pp][0],0)) ||

(is_greater(apot[kk][0][ll][0],apot[kk][0][pp][0],0) &&

is_greater(apot[kk][0][ll][1],apot[kk][0][pp][1],0) &&


is_strictly_greater(apot[kk][0][pp][1],apot[kk][0][ll][0],0)) ||


(is_strictly_greater(apot[kk][0][pp][0],apot[kk][0][ll][0],0) &&

is_strictly_greater(apot[kk][0][ll][1],apot[kk][0][pp][1],0)) ||

(is_strictly_greater(apot[kk][0][ll][0],apot[kk][0][pp][0],0) &&

is_strictly_greater(apot[kk][0][pp][1],apot[kk][0][ll][1],0)))
                                {
                                        if (apot[kk][0][ll][0]!=apot[kk][0][ll][1]) {
```

```
                                                v1[kk][0][ll] = inter-
val_split(pol[kk][0],apot[kk][0][ll]);
                                        }
                                        else { return apot[kk][0][ll]; }

                                        if (apot[kk][0][pp][0]!=apot[kk][0][pp][1]) {
                                                v1[kk][0][pp] = inter-
val_split(pol[kk][0],apot[kk][0][pp]);
                                        }
                                        else { return apot[kk][0][pp]; }
                                }
                        }
                }
        }
        return apot;
}
gen interval_overlap(gen pol, gen apot){
        int jj,kk,ll,pp;

        //---Cases intervals overlap: Let p1 have a root in ]a, b[
        //---                        and p2 have a root in ]c, d[.

        //---The intervals ]a, b[ and ]c, d[ overlap if::

        //---1. a<=c &&  b<=d && b>c
        //---OR
        //---2. a>=c && b>=d && d>a
        //---OR
        //---3. a<c && b>d
        //---OR
        //---4. a>c && b<d

        vecteur &v1 = *apot._VECTptr;

        for(kk=0; kk<_size(apot,0).val; kk++) {    //---for each list of the form
[[[a,a],[a,b]...],m]
                for(ll=0; ll<_size(apot[kk][0],0).val; ll++) {   //---for each list of
the form [[a,a],[a,b]...]
                        for(jj=kk+1; jj<_size(apot,0).val; jj++) {
                                for(pp=0; pp<_size(apot[jj][0],0).val; pp++) {

        while((is_greater(apot[jj][0][pp][0],apot[kk][0][ll][0],0) &&

is_greater(apot[jj][0][pp][1],apot[kk][0][ll][1],0) &&

is_strictly_greater(apot[kk][0][ll][1],apot[jj][0][pp][0],0)) ||

(is_greater(apot[kk][0][ll][0],apot[jj][0][pp][0],0) &&

is_greater(apot[kk][0][ll][1],apot[jj][0][pp][1],0) &&


is_strictly_greater(apot[jj][0][pp][1],apot[kk][0][ll][0],0)) ||


(is_strictly_greater(apot[jj][0][pp][0],apot[kk][0][ll][0],0) &&

is_strictly_greater(apot[kk][0][ll][1],apot[jj][0][pp][1],0)) ||

(is_strictly_greater(apot[kk][0][ll][0],apot[jj][0][pp][0],0) &&

is_strictly_greater(apot[jj][0][pp][1],apot[kk][0][ll][1],0)))
                                        {
                                                v1[kk][0][ll] = inter-
val_split(pol[kk][0],v1[kk][0][ll]);
```

```
                                                v1[jj][0][pp] = inter-
val_split(pol[jj][0],v1[jj][0][pp]);
                                        }
                                }
                        }
                }
        }
        return apot;
}
gen interval_sort(gen apot){
        //---sort the intervals for output

        gen apot2, union_SET, sorted_SET;

        //---change slightly the format of apot, which currently is of the form
        //---[[[[a1,b1],[a2,b2],...[ak,bk]],m1],[[[,],[,],...[,]],m2],...]
        //---and make it so that now EACH interval is, associated with its root multi-
plicity
        //---i.e.
[[[[a1,b1],m1],[[a2,b2],m1]...[[ak,bk],m1]],[[[,],m2],[[,],m2]...[[,],m2]],...]

        apot2 = _map(makevecteur(apot,
                                                        gen("B->if (length(head(B)) == 1)
{[[head(head(B)),B[1]]]} else {map(head(B),C->[C,B[1]])}",0)),0);

        //---union the above lists (make them one set)
        union_SET = gen("set[]",0);

        for (int j_ind=0; j_ind<_size(apot2,0).val; j_ind++) {
                union_SET =
_convert(makevecteur(_union(makevecteur(union_SET,apot2[j_ind]),0),gen("list",0)),0);
        }

        //---sort the set and return it converted back to list again
        sorted_SET = _sort(makevecteur(union_SET,
                                                                gen("(x,y)-
>when(head(x)[0]==head(y)[0],head(x)[1]<head(y)[1],head(x)[0]<head(y)[0])",0)),0);

        return(_convert(makevecteur(sorted_SET,gen("list",0)),0));
}
gen variations(gen f, gen x){
        gen le,c;
        int vars = 0;

        c = _select(makevecteur(gen("x->x!=0",0),_symb2poly(f,0)),0);
        le = _size(c,0);

        if (is_strictly_greater(le,gen(1,0),0)) {
                for(int j_ind=0; j_ind<=le.val-2; j_ind++){
                        if (is_strictly_greater(gen(0,0),c[j_ind]*c[j_ind+1],0)) {
vars++; }
                }
        }

        return gen(vars,0);
}
gen posubLMQ(gen p, gen x){
        //---implements the Local_Max_Quadratic method (LMQ) to compute an
        //---upper bound on the values of the POSITIVE roots of p(x).

        //---Reference:"Linear and Quadratic Complexity Bounds on the Values of the
        //---Positive Roots of Polynomials" by Alkiviadis G. Akritas.
        //---Journal of Universal Computer Science, Vol. 15, No. 3, 523-537, 2009.

        //---Replaced 10^100000 by +infinity
```

```
        gen cl,q,timesused,tempmin,tempmax;
        int len;

        cl = _coeff(p,0);
        q = gen(-1,0);
        tempmax = gen("0.0",0);

        if (_head(cl,0).val<0) { cl = -cl; }
        cl = _revlist(cl,0);
        len = _size(cl,0).val;

        if (len <= 1) { return gen(0,0); }

        timesused = _makelist(makevecteur(gen(1,0),gen(1,0),gen(len,0)),0);
        vecteur &v1 = *timesused._VECTptr;

        for(int m=len-1; m>=1; m--){
                if (is_strictly_greater(gen(0,0),cl[m-1],0)) {
                        tempmin = gen("+infinity",0);
                        for(int n=len; n>= m+1; n--){
                                if (is_strictly_greater(cl[n-1],gen(0,0),0)) {
                                        // form the expression (2^(timesused[n-1])*(-
cl[m-1]/cl[n-1]))^(1/(n-m))
                                        gen a1 = _pow(makevecteur(gen(2,0),timesused[n-
1]),0);  // 2^(timesused[n-1])
                                        gen a2 = -cl[m-1]/cl[n-1];
// (-cl[m-1]/cl[n-1])
                                        gen a3 = a1*a2;
                                        gen a4 = gen(1,0)/gen(n-m,0);
                                        gen a5 = _pow(makevecteur(a3,a4),0);

                                        q = _evalf(a5,0);
                                        v1[n-1] = v1[n-1]+1;

                                        if(is_strictly_greater(tempmin,q,0)) { tempmin =
q; }
                                }
                        }
                        if(is_strictly_greater(tempmin,tempmax,0)) { tempmax = tempmin;
}
                }
        }
        return _ceil((65/64)*tempmax,0);
}
gen poslbdLMQ(gen p, gen x){
        //---implements the Local_Max_Quadratic method (LMQ) to compute a
        //---lower bound on the values of the POSITIVE roots of p(x).

        //---Reference:"Linear and Quadratic Complexity Bounds on the Values of the
        //---Positive Roots of Polynomials" by Alkiviadis G. Akritas.
        //---Journal of Universal Computer Science, Vol. 15, No. 3, 523-537, 2009.

        //---Replaced 10^100000 by +infinity

        gen cl,q,timesused,tempmin,tempmax;
        int len;

        cl = _revlist(_coeff(p,0),0);
        q = gen(-1,0);
        tempmax = gen("0.0",0);

        if (_head(cl,0).val<0) { cl = -cl; }
        len = _size(cl,0).val;
```

```
        if (len <= 1) { return gen(0,0); }

        timesused = _makelist(makevecteur(gen(1,0),gen(1,0),gen(len,0)),0);
        vecteur &v1 = *timesused._VECTptr;

        for(int m=1; m<=len-1; m++) {
                if (is_strictly_greater(gen(0,0),cl[m],0)) {
                        tempmin = gen("+infinity",0);
                        for(int n=0; n<=m-1; n++) {
                                if (is_strictly_greater(cl[n],gen(0,0),0)) {
                                        // form the expression (2^(timesused[n])*(-
cl[m]/cl[n]))^(1/(m-n))
                                        gen a1 =
_pow(makevecteur(gen(2,0),timesused[n]),0);  // 2^(timesused[n])
                                        gen a2 = -cl[m]/cl[n];
// (-cl[m]/cl[n])
                                        gen a3 = a1*a2;
                                        gen a4 = gen(1,0)/gen(m-n,0);
                                        gen a5 = _pow(makevecteur(a3,a4),0);

                                        q = _evalf(a5,0);
                                        v1[n] = v1[n]+1;

                                        if(is_strictly_greater(tempmin,q,0)) { tempmin =
q; }
                                }
                        }
                        if(is_strictly_greater(tempmin,tempmax,0)) { tempmax = tempmin;
}
                }
        }
        // 2^(-ceiling(logb(tempmax,2)))
        gen ret = _pow(makevecteur(gen(2,0),-
_ceil(_logb(makevecteur(tempmax,gen(2,0)),0),0)),0);
        return ret;
}
gen intrv(gen a, gen b){
        if (is_strictly_greater(a,b,0)) { return gen(makevecteur(b,a),0); }
        else { return gen(makevecteur(a,b),0); }
}
gen VAS_Pos_Roots(gen p, gen x, gen ap, gen bp, gen cp, gen dp){
        //---The steps below correspond to the steps described in the reference below.

        //---Reference:        "A Comparative Study of Two Real Root Isolation Methods"
        //---by Alkiviadis G. Akritas and Adam W. Strzebonski.
        //---Nonlinear Analysis: Modelling and Control, Vol. 10, No. 4, 297-304, 2005.

        gen f = p;
        gen intervalsToBeProcessed("[]",0);
        gen rootIsolationIntervals("[]",0);
        gen a,b,c,d;
        gen a1,b1,c1,d1;
        gen a2,b2,c2,d2;
        gen att,bt,ct,dt;
        gen vrs,vrs1,vrs2,vrst,ub,lb,f1,f2,ft,r;
        gen xx,yy;

        //step 1//

        vrs = variations(f,x);
        ub = posubLMQ(f,x);

        if (vrs.val == 0) { return (rootIsolationIntervals); }
        if (vrs.val == 1) {
```

```
                rootIsolationIntervals =
_prepend(makevecteur(rootIsolationIntervals,intrv(0,ap*ub)),0);
                return rootIsolationIntervals;
        }
        intervalsToBeProcessed =
        _prepend(makevecteur(intervalsToBeProcessed,gen(makevecteur(ap,bp,cp,dp,f,vrs),
0)),0);

        //step 2//

        while (_size(intervalsToBeProcessed,0).val != 0) {
                gen l1 = _head(intervalsToBeProcessed,0);
                a = l1[0];
                b = l1[1];
                c = l1[2];
                d = l1[3];
                f = l1[4];
                vrs = l1[5];
                intervalsToBeProcessed = _tail(intervalsToBeProcessed,0);

                //step 3//

                lb = poslbdLMQ(f,x);

                //step 4//

                if (is_strictly_greater(lb,gen(16,0),0)) {
                        // form expression "x=lb*x"
                        string s = "x=" + print(lb,0) + "*x";
                        gen expr1(s,0);

                        f = expand(_subst(makevecteur(f,expr1),0),0);
                        a = lb*a;
                        c =lb*c;
                        lb = gen(1,0);
                }

                //step 5//

                if (is_greater(lb,gen(1,0),0)) {
                        // form expression "x=lb+x"
                        string s = "x=" + print(lb,0) + "+x";
                        gen expr1(s,0);

                        f = expand(_subst(makevecteur(f,expr1),0),0);
                        b = lb*a + b;
                        d = lb*c + d;

                        if (_subst(makevecteur(f,gen("x=0",0)),0)==0) {
                                rootIsolationIntervals =
_append(makevecteur(rootIsolationIntervals,makevecteur(b/d,b/d)),0);
                                f = giac::normal(f/x,0);
                        }
                        vrs = variations(f,x);

                        if (vrs.val == 0) { continue; }
                        if (vrs.val == 1) {
                                if (c.val != 0) {
                                        rootIsolationIntervals =
_append(makevecteur(rootIsolationIntervals,intrv(a/c,b/d)),0);
                                }
                                else {
                                        rootIsolationIntervals =
_append(makevecteur(rootIsolationIntervals,
```

```
                                        intrv(b,b+a*posubLMQ(f,x))),0);
                                }
                                continue;
                        }
                }

                //step 6//

                f1 = expand(_subst(makevecteur(f,gen("x=x+1",0)),0),0);
                a1 = a;
                b1 = a + b;
                c1 = c;
                d1 = c + d;
                r = 0;

                if (_subst(makevecteur(f1,gen("x=0",0)),0)==0) {
                        rootIsolationIntervals =
_append(makevecteur(rootIsolationIntervals,makevecteur(b1/d1,b1/d1)),0);
                        f1 = giac::normal(f1/x,0);
                        r = 1;
            }

                vrs1 = variations(f1,x);
                vrs2 = vrs - vrs1 - r;
                a2 = b;
                b2 = a + b;
                c2 = d;
                d2 = c + d;

                //step 7//

                if (vrs2.val > 1) {
                        f2 = ex-
pand(_subst(makevecteur(_poly2symb(_revlist(_symb2poly(f,0),0),0),gen("x=x+1",0)),0),0
);
                        if (_subst(makevecteur(f2,gen("x=0",0)),0)==0) { f2 =
giac::normal(f2/x,0); }
                        vrs2 = variations(f2,x);
                }

                //step 8//

                if (vrs1.val < vrs2.val) {
                        att = a1;  bt = b1;  ct = c1;  dt = d1;  ft = f1;  vrst = vrs1;
                        a1 = a2;   b1 = b2;  c1 = c2;  d1 = d2;  f1 = f2;  vrs1 = vrs2;
                        a2 = att;  b2 = bt;  c2 = ct;  d2 = dt;  f2 = ft;  vrs2 = vrst;
                }

                //step 9//

                if (vrs1.val == 0) { continue; }
                if (vrs1.val == 1) {
                        if (c1.val != 0) {
                                rootIsolationIntervals =
_append(makevecteur(rootIsolationIntervals,intrv(a1/c1,b1/d1)),0);
                        }
                        else {
                                rootIsolationIntervals =
_append(makevecteur(rootIsolationIntervals,

                                 intrv(b1,b1+a1*posubLMQ(f1,x))),0);
                        }
                }
                else {
```

```
                        intervalsToBeProcessed =

        _prepend(makevecteur(intervalsToBeProcessed,makevecteur(a1,b1,c1,d1,f1,vrs1)),0
);
                }

                //step 10//

                if (vrs2.val == 0) { continue; }
                if (vrs2.val == 1) {
                        if (c2.val != 0) {
                                rootIsolationIntervals =

        _append(makevecteur(rootIsolationIntervals,intrv(a2/c2,b2/d2)),0);
                        }
                        else {
                                rootIsolationIntervals =
_append(makevecteur(rootIsolationIntervals,

                                 intrv(b2,b2+a2*posubLMQ(f2,x))),0);
                        }
                }
                else {
                        intervalsToBeProcessed =

        _prepend(makevecteur(intervalsToBeProcessed,makevecteur(a2,b2,c2,d2,f2,vrs2)),0
);
                }
                rootIsolationIntervals =
_revlist(_sort(makevecteur(rootIsolationIntervals,

                                gen("(xx,yy)-
>when(xx[1]==yy[1],xx[0]>yy[0],xx[1]>yy[1])",0)),0),0);
        }
        return rootIsolationIntervals;
}
gen VAS(gen p, gen x, gen ap, gen bp, gen cp, gen dp){
        //---calls VAS_Pos_Roots to compute the positive and negative roots (checking
for zero as well)
        //---and forms the list of isolating intervals of a poly WITHOUT multiple
roots.

        gen a,a1,pp,apot;
        int k;

    a = VAS_Pos_Roots(p,x,ap,bp,cp,dp);   //---isolate the positive roots

        gen tmp = _poly2symb(_symb2poly(p,0),0);
        tmp = _subst(makevecteur(tmp,gen("x=-x",0)),0);
        pp = _simplify(_pow(makevecteur(gen(-1,0),_degree(p,0)),0)*tmp,0);

        //---if poly is symmetric just copy intervals
        if (p == pp) { a1 = _revlist(_map(makevecteur(-a,gen("sort",0)),0),0); }

        //---else isolate the negative roots
        else { a1 = VAS_Pos_Roots(pp,x,-ap,bp,cp,dp); }

        k = 0;

        if(_horner(makevecteur(p,0),0)==0) {
                apot =
_makelist(makevecteur(gen(0,0),gen(1,0),gen(_size(a1,0).val+_size(a,0).val+1,0)),0);
                vecteur &vct = *apot._VECTptr;
                while(k<_size(a1,0).val) {
                        vct[k] = a1[k];
```

```
                        k++;
                }
                vct[k] = makevecteur(gen(0,0),gen(0,0));
                k++;

                while(k<=_size(a1,0).val+_size(a,0).val) {
                        vct[k] = a[k-_size(a1,0).val-1];
                        k++;
                }
        }
        else {
                apot =
_makelist(makevecteur(gen(0,0),gen(1,0),gen(_size(a1,0).val+_size(a,0).val,0)),0);
                vecteur &vct = *apot._VECTptr;
                while(k<_size(a1,0).val) {
                        vct[k] = a1[k];
                        k++;
                }
                while(k<_size(a1,0).val+_size(a,0).val) {
                        vct[k] = a[k-_size(a1,0).val];
                        k++;
                }
        }

        return apot;
}
gen VAS_Real_Roots_R(gen pt){
        //mpf_set_default_prec(30);

        gen irrationalRootsIntervals("[]",0);
        gen rationalRootsIntervals("[]",0);
        gen allIntervals("[]",0);
        gen rationalRootsONLY(0,0);        //---rationalRootsONLY==1 means there are
ONLY rational roots
        gen listRationalFactors;
        gen productRationalFactors(1,0);
        gen poly,pol;

        //---Check for valid input
        if ((_size(_lname(pt,0),0)).val > 1 || (_size(_lname(pt,0),0)).val == 0) {
                NSLog(@"Invalid input poly; should have one variable.");
        }

        //---change poly variable to x, so that it works for polys in ANY variable.
        gen var = _lname(pt,0)[0];
        gen p = giac::normal(_poly2symb(_symb2poly(makevecteur(pt,var),0),0),0);

        //---continue with the poly in variable x
        //    ...
        //---and with positive leading coefficient
        if ((_lcoeff(p,0).val)<0) { p = giac::normal((-1)*p,0); }

        //---remove the content for smaller coefficients (and for my_sqrfree)
        p = giac::normal(p/_gcd(_symb2poly(p,0),0),0);

        my_sqrfree(p);

        //---find the rational roots and their multiplicity --using the function ra-
tionalroot(p)--
        //---and form their "isolating intervals" of the form [[[a, a], m], ...]

        rationalRootsIntervals = formRationalRootIntervals(p);

        if (_size(rationalRootsIntervals,0).val>1 ||
(_size(rationalRootsIntervals,0).val==1
```

```
         && _size(rationalRootsIntervals[0][0],0).val!= 0)) {
               //---form the product of the factors with rational roots and divide it
out of the poly
               listRationalFactors = _map(makevecteur(rationalRootsIntervals,

gen("y->map(head(y),if (head(y)!=[]){A->(x-A[0])^head(tail(y))})",0)),0);

               listRationalFactors = _remove(makevecteur(gen("x-
>x==[]",0),listRationalFactors),0);

               for (int ind_j=0; ind_j<_size(listRationalFactors,0).val; ind_j++) {
                      productRationalFactors =

       giac::normal(productRationalFactors*_product(listRationalFactors[ind_j],0),0);
               }
       }

       //---if poly == 1, all roots are rational. Set rationalRootsONLY to 1!
       poly = giac::normal(_quo(makevecteur(p,productRationalFactors),0),0);
       if (poly.val == 1) {
               poly = p;
               gen irrationalRootsIntervals("[]",0);
               rationalRootsONLY = gen(1,0);
       }

       //---remove the content for smaller coefficients (and for my_sqrfree)
       //---rationalRootsONLY==0 means we have irrational roots;  isolate them using
VAS
       if (rationalRootsONLY.val==0) {
               poly = giac::normal(poly/_gcd(_symb2poly(poly,0),0),0);
               pol = my_sqrfree(poly);

               gen tmpv;
               for(int k=0; k<_size(pol,0).val; k++) {
                      tmpv = makevec-
teur(VAS(pol[k][0],gen("x",0),gen(1,0),gen(0,0),gen(0,0),gen(1,0)),pol[k][1]);
                      irrationalRootsIntervals =
_append(makevecteur(irrationalRootsIntervals,tmpv),0);
               }
       }

       //---merge the intervals for the rational and irrational roots
       allIntervals =
_union(makevecteur(rationalRootsIntervals,irrationalRootsIntervals),0);

       //---and sort them according to multiplicity
       allIntervals = _sort(makevecteur(allIntervals,gen("(x,y)->x[1]<y[1]",0)),0);

       //---If rationalRootsONLY==0, combine sublists of same multiplicity (to conform
to structure of pol)
       //---allIntervals is now of the form [[[[a,b],[],..],m],...].  Moreover, in
this case,
       //---a rational root [[a,a], m] does NOT have the corresponding factor in pol,
       //---so we insert in it the factor [x-a, m].
       if (rationalRootsONLY.val==0) {
               allIntervals = combineSublistsOfSameMultiplicity(allIntervals);
               pol = addMissingSqrfreeFactors(pol,allIntervals);
       }
       else { pol = my_sqrfree(poly); }

       //---If rationalRootsONLY==0, make sure there is no interval overlap within a
given multiplicity
       //---and among different multiplicities
```

```
        if (rationalRootsONLY.val==0) {
                allIntervals = interval_overlap2(pol,allIntervals);
                allIntervals = interval_overlap(pol,allIntervals);
        }
        //---sort intervals
        allIntervals = interval_sort(allIntervals);

        return allIntervals;
}
gen approximate_Real_Roots_R(gen ff, gen LIST, gen eps, gen maxIter) {
        //---Approximates roots to a given precision,
        //---To be used ONLY when the rational roots are being removed from ff.

        //---The LIST of isolating intervals is returned by VAS_Real_Roots
        //---applied to f;
        //---eps is the desired accuracy of the roots (example: 10^-6);
        //---maxIter is the maximum number of iterations (start with 20).

        //---Comments:: To be used when RATIONAL roots have been computed separately.
        //---This way our functions bisectionMethodR() will NOT deal with intervals [a,
a] and [a, b];
        //---that is, we know for sure that the end-points will NOT be roots of the
poly!!
        //---No need for the machine_Epsilon()!

        //---Plans for the future:: PROVIDE additional choices like Brent's method.

        gen LIST2, LIST3, LIST4, LIST5, rootsLIST, var, f;
        gen rationalRootsIntervals,irrationalRootsIntervals, listRationalFactors,
productRationalFactors;
        gen rationalRootsLIST, irrationalRootsLIST, rootsSET;
        //identificateur x("x");

        rationalRootsIntervals = gen("[0]",0); //---dynamic (mutable list; can grow as
needed)
        irrationalRootsIntervals = gen("[0]",0);
        rootsSET = gen("[0]",0);

        //---change poly variable to x, so that it works for polys in ANY variable.
        var = _lname(ff,0)[0];
        f = giac::normal(_poly2symb(_symb2poly(makevecteur(ff,var),0),0),0);

        //---remove the content for smaller coefficients (and for my_sqrfree2)
        f = giac::normal(f/_gcd(_symb2poly(f,0),0),0);

        //---continue with the poly in variable x
        //---and with positive leading coefficient
        if ((_lcoeff(f,0).val)<0) { f = giac::normal((-1)*f,0); }

        //---get the list of intervals without their multipicities
        //LIST2 = _map(makevecteur(LIST,gen("D->head(D)",0)),0);

        //---Remove from LIST the intervals corresponding to the RATIONAL roots
        //---because those are already approximated (no need to be approximated by bi-
section).
        LIST3 = _map(makevecteur(LIST,gen("x->if (x[0]==x[1]) { x } else {0}",0)),0);
        LIST4 = _remove(makevecteur(gen("x->x==0",0),LIST3),0);
        rationalRootsLIST = _map(makevecteur(LIST4,gen("x->approx(x[0])",0)),0);

        //---form the product of the linear factors with rational roots
        listRationalFactors = _map(makevecteur(LIST4,gen("y->(x-head(y))",0)),0);
        productRationalFactors = giac::normal(_product(listRationalFactors,0),0);

        //---the intervals corresponding to the IRRATIONAL roots are in LIST5.
        //---These need to be approximated with bisection.
```

```
        LIST5 =
_convert(makevecteur(_minus(makevecteur(LIST,LIST4),0),gen("list",0)),0);

        if(_size(LIST5,0).val==0) { return rationalRootsLIST; }

        //---get the product of the square free factors (WITHOUT multiplicities)
        f = my_sqrfree2(f);

        //---divide out the product of linear factors
        f = giac::normal(_quo(makevecteur(f,productRationalFactors),0),0);

        //---poly has only simple irrational roots to be approximated
        //irrationalRootsLIST = _map(makevecteur(LIST5,gen(s1,0)),0);

        vecteur &v1 = *LIST5._VECTptr;
        // irrationalRootsLIST:=map(LIST5,E-
>bisectionMethod_R(f,x,head(E),head(tail(E)),eps,maxIter));
        for(int i=0; i<_size(LIST5,0).val; i++) {
                v1[i] = bisection-
Method_R(f,gen("x",0),_head(v1[i],0),_head(_tail(v1[i],0),0),eps,maxIter);
        }

        irrationalRootsLIST = LIST5;

        //---merge the two lists of roots
        rootsSET = _union(makevecteur(rationalRootsLIST,irrationalRootsLIST),0);

        return _sort(_convert(makevecteur(rootsSET,gen("list",0)),0),0);
}

- (BOOL)getIntervals:(NSString *)inputPoly with:(NSString *)method andPrecisionDig-
its:(int)precDigits{
        /*
        gen pt("x^200-2*(5*x-1)^2",0);
        vecteur &v1 = *appr._VECTptr;
        */
        // time 'VAS_Real_Roots_R()' and assign the result in 'computationTime' proper-
ty

        gen p(string([inputPoly UTF8String]),0);   // convert inputPoly (NSString) to
gen (p)

        if(isPoly(p)==-1) { return NO; }

        // if input poly contains more than 1 variable, return NO
        if(_size(_lname(p,0),0).val>1)        { return NO; }

        init_time();
        int64 begin,end;
        gen allIntervals;

        if ([method isEqual:@"VAS"]) {
                p =
giac::normal(_poly2symb(_symb2poly(makevecteur(p,_lname(p,0)[0]),0),0),0);

                // take the numerator if poly is (x-a/b)*(x-c/d)*...*(x-y/z)
                p = _numer(_simplify(p,0),0);
                p = _simplify(p,0);

                begin = get_time1();
                //allIntervals = VAS_Real_Roots_R(p);
                //allIntervals = _VAS(p,0);
                allIntervals = _realroot(p,0);
         end = get_time1();
```

```
        for(int i=0; i<_size(allIntervals,0).val; i++) {
                        vecteur &v1 = *allIntervals[i]._VECTptr;

                        // turn roots to intervals (e.g. [3,1]->[[3,3],1])
                        if(v1[0].type != _VECT) {
                                v1[0] = makevecteur(gen(v1[0]),gen(v1[0]));
                        }
                }
    }
        else if ([method isEqual:@"Sturm"]) {
                // take the numerator if poly is (x-a/b)*(x-c/d)*...*(x-y/z)
                p = _numer(_simplify(p,0),0);
                p = _simplify(p,0);

                begin = get_time1();
                allIntervals = _realroot(makevecteur(gen("sturm",0),p),0);
                end = get_time1();

                allIntervals = _evalf(allIntervals,0);

                for(int i=0; i<_size(allIntervals,0).val; i++) {
                        vecteur &v1 = *allIntervals[i]._VECTptr;

                        // turn roots to intervals
                        if(v1[0].type != _VECT) {
                                v1[0] = makevecteur(gen(v1[0]),gen(v1[0]));
                        }
                        v1[1] = gen((int)v1[1]._DOUBLE_val,0);
                }

                for(int i=0; i<_size(allIntervals,0).val; i++) {
                        vecteur &v1 = *allIntervals[i]._VECTptr;
                        vecteur &v2 = *v1[0]._VECTptr;

                        // remove ".0" from exact roots
                        string doubleRoot;
                        int len;

                        for (int j1=0; j1<2; j1++) {
                                doubleRoot = print(v2[j1],0);
                                len = doubleRoot.length();

                                if (doubleRoot.substr(len-2,len-1)==".0") { v2[j1] =
gen(doubleRoot.substr(0,len-2),0); }
                                else {
                                        // truncate doubles to set precision
                                        if(doubleRoot[0]=='-') { v2[j1] =
gen(doubleRoot.substr(0,precDigits+3),0); }
                                        else { v2[j1] =
gen(doubleRoot.substr(0,precDigits+2),0); }
                                }
                        }
                }
        }

        self.intervals = [[NSString alloc] initWith-
UTF8String:print(allIntervals,0).c_str()];
        self.intervals = [self.intervals stringByReplacingOccurrencesOfString:@"],["
withString:@"], ["];
        self.computationTime = [NSString stringWithFormat:@"%llf",(end-
begin)/pow(10.0,6.0)];

        // get 'normal' form of inputPoly and set 'polynomial' property with the corre-
sponding NSString
        p = giac::normal(p,0);
```

```
        self.polynomial = [[NSString alloc] initWithUTF8String:print(p,0).c_str()];

        return YES;
}

- (BOOL)getRoots:(NSString *)inputPoly withPrecisionDigits:(int)precDigits {

        gen p(string([inputPoly UTF8String]),0);   // convert inputPoly (NSString) to
gen (p)

        if(isPoly(p)==-1) { return NO; }

        // if input poly contains more than 1 variable, return NO
        if(_size(_lname(p,0),0).val>1)        { return NO; }

        // take the numerator if poly is (x-a/b)*(x-c/d)*...*(x-y/z)
        p = _numer(_simplify(p,0),0);
        p = _simplify(p,0);

        //p = expand(_poly2symb(_numer(p,0),0),0);
        //p = giac::normal(p,0);

        //gen allIntervals = VAS_Real_Roots_R(p);
        gen allIntervals = _VAS(p,0);

        vecteur &v0 = *allIntervals._VECTptr;

        for(int i=0; i<_size(allIntervals,0).val; i++) {
                if (v0[i].type != _VECT) { v0[i] = makevecteur(gen(v0[i]),gen(v0[i]));
}
        }

        gen appr = approximate_Real_Roots_R(p, allIntervals,
_pow(makevecteur(gen(10,0),gen(-6,0)),0), gen(20,0));

        vecteur &v1 = *appr._VECTptr;
        string rootsStr = "[";
        for(int i=0; i<_size(appr,0).val; i++) {
                if(print(v1[i],0)[0]=='-') { rootsStr = rootsStr +
print(v1[i],0).substr(0,precDigits+3)+", "; }
                else {
                    rootsStr = rootsStr + print(v1[i],0).substr(0,precDigits+2)+", ";
                }
        }
        rootsStr = rootsStr.substr(0,rootsStr.length()-2) + "]";

        self.rootList = [[NSString alloc] initWithUTF8String:rootsStr.c_str()];

        // get 'normal' form of inputPoly and set 'polynomial' property with the corre-
sponding NSString
        p = giac::normal(p,0);
        self.polynomial = [[NSString alloc] initWithUTF8String:print(p,0).c_str()];

        return YES;
}
- (BOOL)getRationalRoots:(NSString *)inputPoly {
        gen p(string([inputPoly UTF8String]),0);   // convert inputPoly (NSString) to
gen (p)

        if(isPoly(p)==-1) { return NO; }

        // if input poly contains more than 1 variable, return NO
        if(_size(_lname(p,0),0).val>1)        { return NO; }

        // take the numerator if poly is (x-a/b)*(x-c/d)*...*(x-y/z)
```

```
        p = _numer(_simplify(p,0),0);
        p = _simplify(p,0);

        gen rationalRoots = _rationalroot(p,0);

        self.rootList = [[NSString alloc] initWith-
UTF8String:print(rationalRoots,0).c_str()];
        self.rootList = [self.rootList stringByReplacingOccurrencesOfString:@"," with-
String:@", "];

        // get 'normal' form of inputPoly and set 'polynomial' property with the corre-
sponding NSString
        p = giac::normal(p,0);
        self.polynomial = [[NSString alloc] initWithUTF8String:print(p,0).c_str()];

        return YES;
}
- (BOOL)getBound:(NSString *)inputPoly kind:(NSString *)kindOfBound  {
        gen p(string([inputPoly UTF8String]),0);   // convert inputPoly (NSString) to
gen (p)

        // if input poly contains more than 1 variable, return NO
        if(_size(_lname(p,0),0).val>1)        { return NO; }

        p = giac::normal(p,0);
        self.polynomial = [[NSString alloc] initWithUTF8String:print(p,0).c_str()];

        if(isPoly(p)==-1) { return NO; }

        // take the numerator if poly is (x-a/b)*(x-c/d)*...*(x-y/z)
        p = _numer(_simplify(p,0),0);
        p = _simplify(p,0);

        self.polynomial = [[NSString alloc] initWithUTF8String:print(p,0).c_str()];

        if ([kindOfBound isEqual:@"upper"]) {
                //gen upperBound = posubLMQ(p,gen("x",0));
                gen upperBound = _posubLMQ(p,0);

                if (print(upperBound,0)=="+infinity") { self.upperBoundStr = @"+∞"; }
         else { self.upperBoundStr = [[NSString alloc] initWith-
UTF8String:print(upperBound,0).c_str()]; }
        }
        else if ([kindOfBound isEqual:@"lower"]) {
                //gen lowerBound = poslbdLMQ(p,gen("x",0));
                gen lowerBound = _poslbdLMQ(p,0);

                if (print(lowerBound,0)=="+infinity") { self.lowerBoundStr = @"+∞"; }
                else { self.lowerBoundStr = [[NSString alloc] initWith-
UTF8String:print(lowerBound,0).c_str()]; }
        }
        return YES;
}
- (BOOL)getFactoredPoly:(NSString *)inputPoly kind:(NSString *)kindOfFactorization {
        gen p(string([inputPoly UTF8String]),0);   // convert inputPoly (NSString) to
gen (p)

        // if input poly contains more than 1 variable, return NO
        if(_size(_lname(p,0),0).val>1)        { return NO; }

        p = giac::normal(p,0);
        self.polynomial = [[NSString alloc] initWithUTF8String:print(p,0).c_str()];

        if(isPoly(p)==-1) { return NO; }
```

```
        // take the numerator if poly is (x-a/b)*(x-c/d)*...*(x-y/z)
        p = _numer(_simplify(p,0),0);
        p = _simplify(p,0);

        self.polynomial = [[NSString alloc] initWithUTF8String:print(p,0).c_str()];

        if ([kindOfFactorization isEqual:@"normalFactorization"]) {
                p = _factor(p,0);
                self.factoredPolyStr = [[NSString alloc] initWith-
UTF8String:print(p,0).c_str()];
        }
        else if ([kindOfFactorization isEqual:@"sqrFreeFactorization"]) {
                p = _sqrfree(p,0);
                self.factoredSqrFreePolyStr = [[NSString alloc] initWith-
UTF8String:print(p,0).c_str()];
        }
        return YES;
}
+ (NSString *)getNormalPoly:(NSString *)inputPoly {
        gen p(string([inputPoly UTF8String]),0);   // convert inputPoly (NSString) to
gen (p)
        p = giac::normal(p,0);

        if(isPoly(p)==-1) { return @"not a poly"; }

        // take the numerator if poly is (x-a/b)*(x-c/d)*...*(x-y/z)
        p = _numer(_simplify(p,0),0);
        p = _simplify(p,0);

        // if input poly contains more than 1 variable, return NO
        if(_size(_lname(p,0),0).val>1)        { return @"not a poly"; }

        NSString *polyStr1 = [[NSString alloc] initWithUTF8String:print(p,0).c_str()];
        [polyStr1 autorelease];
        return [polyStr1 stringByReplacingOccurrencesOfString:@"*" withString:@""];
}

- (void)dealloc {
        self.rootList = nil;
        self.intervals = nil;
        self.polynomial = nil;
        self.lowerBoundStr = nil;
        self.upperBoundStr = nil;
        self.factoredPolyStr = nil;
        self.factoredSqrFreePolyStr = nil;
        [super dealloc];
}

@end
```

Αναφορές & Βιβλιογραφία

[1] A. G. Akritas, *Vincent's theorem in algebraic manipulation.* Ph.D. Thesis, Operations Research Program, North Carolina State University, Raleigh, NC, (1978).

[2] A. G. Akritas, *An implementation of Vincent's Theorem.* – Numerische Mathematik **36**, (1980), 53-62.

[3] A. G. Akritas, *The fastest exact algorithms for the isolation of the real roots of a polynomial equation.* – Computing **24** (1980), 299-313.

[4] A. G. Akritas, *Reflections on a pair of theorems by Budan and Fourier.* – Mathematics Magazine **55**, 5 (1982), 292-298.

[5] A. G. Akritas, *Elements of Computer Algebra with Applications.* John Wiley Interscience, New York, 1989.

[6] A. G. Akritas, *Linear and quadratic complexity bounds on the values of the positive roots of polynomials.*

[7] A. G. Akritas, Vincent's Theorem of 1836: Overview and Future Research. In N. N. Vassiliev (Ed.), /Proceedings of the International Conference on Polynomial Computer Algebra/, PCA 2009, 48-51, St. Petersburg, Russia, April 8-12, 2009. Euler International Mathematical Institute, Russian Academy of Sciences.

[8] A. G. Akritas, A. I. Argyris, A. W. Strzebonski, *FLQ, the Fastest Quadratic Complexity Bound on the Values of Positive Roots of Polynomials.* – Serdica Journal of Computing **2**, (2008), 145-162.

[9] A. G. Akritas, A. Bocharov, A. W. Strzebonski, *Implementation of real root isolation algorithms in Mathematica.* – Abstracts of the International Conference on Interval and Computer-Algebraic Methods in Science and Engineering (Interval '94), St. Petersburg, Russia, (March 7-10), (1994), 23-27.

[10] A. G. Akritas, A. Strzebonski, *A comparative study of two real root isolation methods.* – Nonlinear Analysis: Modelling and Control **10**, 4 (2005), 297-304.

[11] A. G. Akritas, P. Vigklas, *A Comparison of Various Methods for Computing Bounds for Positive Roots of Polynomials.* – Journal of Universal Computer Science **13**, 4 (2007), 455-467.

[12] A. G. Akritas, A. Strzebonski, P. Vigklas, *Implementations of a New Theorem for Computing Bounds for Positive Roots of Polynomials.* – Computing 78 (2006), 355-367.

[13] A. G. Akritas, A. Strzebonski, P. Vigklas, *Improving the Performance of the Continued Fractions Method Using new Bounds of Positive Roots.* – Nonlinear Analysis: Modelling and Control **13** (3) (2008), 265-279.

[14] A. Alesina, M. Galuzzi, *A new proof of Vincent's theorem.* – L'Enseignement Mathematique **44** (1998), 219-256.

[15] F. Boulier, *Systèmes polynomiaux: que signifie "résoudre"?.* – Lect.Notes, Université Lille 1, 8 janvier 2007.

[16] G. E. Collins, A. G. Akritas, *Polynomial real root isolation using Descartes' rule of signs.* – In: Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computations, Yorktown Heights, N.Y., (1976), pp. 272-275.

[17] H. Hong, *Bounds for absolute positiveness of multivariate polynomials.* – J. Symb. Comput. **25** (5) (1998), 571-585.

[18] J. von zur Gathen, J. Gerhard, *Fast Algorithms for Taylor Shifts and Certain Difference Equations.* – In: Proceedings of ISSAC'97, Maui, Hawaii, U.S.A., (1997), pp.40-47.

[19] B. Kioustelidis, *Bounds for positive roots of polynomials.* – J. Comput. Appl. Math. Bf16, 2 (1986), 241-244.

[20] N. Obreschkoff, *Verteilung und Berechnung der Nullstellen reeller Polynome.* – VEB Deutscher Verlag der Wissenschaften, Berlin, (1963)[1].

[21] F. Rouillier, P. Zimmermann, *Efficient isolation of polynomials' real roots.* - Journal of Computational and Applied Mathematics **162** (2004), 33-50.

[22] V. Sharma, *Complexity of Real Root Isolation Using Continued Fractions.* – ISAAC07 preprint (2007).

[23] V. Sharma, *Complexity Analysis of Algorithms in Algebraic Computation.* Ph.D. Thesis, Department of Computer Sciences, Courant Institute of Mathematical Sciences, New York University, 2007.

[24] D. Ştefănescu, *New bounds for positive roots of polynomials.* – Journal of Universal Computer Science **11** (12) (2005), 2132-2141.

[25] E.P. Tsigaridas, I. Z. Emiris, *Univariate polynomial real root isolation: Continued fractions revisited.* – (Y. Azar and T. Erlebach (Eds.)), ESA 2006, LNCS **4168** (2006), 817-828.

[26] A. J. H. Vincent, *Sur la resolution des équations numériques.* – Journal de Mathématiques Pures et Appliquées 1 (1836), 341-372.

[27] Αλκιβιάδης Γ. Ακρίτας, *Υπολογιστική Άλγεβρα με το Mathematica, Τόμος ΙΙ, Ακαδημαικό Έτος 2007-2008, Πανεπιστημιακές Εκδόσεις Θεσσαλίας.*

[28] Apple Developer Tools presentation: http://developer.apple.com/technologies/tools/

[29] Giac/Xcas site: http://www-fourier.ujf-grenoble.fr/~parisse/giac.html

[1] Για μια αγγλική μετάφραση ενός βιβλίου με παρόμοιο περιεχόμενο, βλέπε: Obreschkoff, N.: "Zeros of Polynomials", Bulgarian Academic Monographs (7), Sofia, 2003.

[30] Stanford CS193p, Developing Applications for iOS, Fall 2010, Lecture slides. http://www.stanford.edu/class/cs193p/cgi-bin/drupal/downloads-2010-fall

[31] Wikipedia. iOS article: http://en.wikipedia.org/wiki/IOS

[32] Wikipedia. iOS SDK article: http://en.wikipedia.org/wiki/IOS_SDK