ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Δημιουργία εφαρμογής για Γραμμική Άλγεβρα για iOS

Creation of Linear Algebra application for iOS

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΧΑΣΑΚΟΣ ΣΠΥΡΙΔΩΝ

Βόλος, 2012

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Δημιουργία εφαρμογής για Γραμμική Άλγεβρα για iOS

Creation of Linear Algebra application for iOS

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΧΑΣΑΚΟΣ ΣΠΥΡΙΔΩΝ

Επιβλέποντες:

Αλκιβιάδης Γ. Ακρίτας
Αναπληρωτής Καθηγητής Π.Θ.

Ασπασία Δασκαλοπούλου
Επίκουρος Καθηγήτρια Π.Θ.

Εγκρίθηκε από την διμελή εξεταστική επιτροπή τον Ιούλιο 2012

(Υπογραφή)

….....

Αλκιβιάδης Γ. Ακρίτας

Αναπληρωτής Καθηγητής Π.Θ.

(Υπογραφή)

….....

Αλκιβιάδης Γ. Ακρίτας

Αναπληρωτής Καθηγητής Π.Θ.

Βόλος, 2012

(Υπογραφή)

.....

ΧΑΣΑΚΟΣ ΣΠΥΡΙΔΩΝ

Διπλωματούχος Μηχανικός Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων Πανεπιστημίου Θεσσαλίας

Αφιέρωση

Στην οικογένειά μου και στους φίλους μου

Ευχαριστίες

Αρχικά, ευχαριστώ θερμά τον επιβλέποντα καθηγητή μου κύριο Αλκιβιάδη Ακρίτα για την εμπιστοσύνη που έδειξε στο πρόσωπό μου, την αμέριστη βοήθεια, καθοδήγηση και υποστήριξή του σε κάθε φάση εκπόνησης της παρούσας διπλωματικής, καθώς και για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον και σύγχρονο θέμα.

Ευχαριστώ πολύ την καθηγήτριά μου Ασπασία Δασκαλοπούλου για την τιμή που μου έκανε να αναλάβει την επίβλεψη της διπλωματικής μου εργασίας.

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 Κεχαγιά Σπυρίδωνα και Τζεβελεκίδη Κωνσταντίνο για τη βοήθεια και υποστήριξή τους τόσο κατά τη διάρκεια των σπουδών μου όσο και σε θέματα με την παρούσα διπλωματική εργασία.

Χασακός Σπυρίδων

Βόλος, 2012

Περίληψη

Σκοπός αυτής της εργασίας είναι η παρουσίαση της iOS εφαρμογής “Linear Algebra”, η οποία πραγματεύεται λειτουργίες της γραμμικής άλγεβρας που αφορούν διανύσματα, πίνακες και εξισώσεις.

Αρχικά, αναφέρουμε παρόμοιες εφαρμογές που υπάρχουν για iOS αλλά και για το λειτουργικό Android, εξετάζοντας τις διαφορές και τις ομοιότητες ως προς τη δική μας εφαρμογή. Ακολούθως, παρουσιάζουμε τα εφόδια και τις γνώσεις που απαιτείται να διαθέτει κανείς για να ασχοληθεί με την ανάπτυξη εφαρμογών για iOS.

Στη συνέχεια, περνώντας από το θεωρητικό σκέλος της παρούσας εργασίας σε αυτό που είναι σχετικό με τον προγραμματισμό, εξετάζουμε θέματα σχετικά με την υλοποίηση των λειτουργιών της γραμμικής άλγεβρας 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 βιβλιοθήκης Giac, την οποία χρησιμοποιούμε για τις πράξεις γραμμικής άλγεβρας. Τέλος, παρουσιάζουμε αναλυτικά τις λειτουργίες και τη διεπαφή της εφαρμογής μας, ενώ παραθέτουμε πώς μπορεί να επεκταθεί και να βελτιωθεί η εφαρμογή μας στο μέλλον.

Λέξεις Κλειδιά:

γραμμική άλγεβρα, iOS εφαρμογή, Giac

Abstract

The purpose of this work is to present the iOS application “Linear Algebra”, which deals with the linear algebra operations on vectors, tables and equations.

Initially, we mention similar applications for iOS and for the operating system Android, examining the differences and similarities compared with our application. Subsequently, we present the skills and knowledge one must possess in order to develop applications for iOS.

Moving from the theoretical part of this thesis to the programming – related, we examine issues concerning the implementation of the linear algebra operations 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 linear algebra operations. Finally, we thoroughly present the features and interface of our application, while we cite future improvements and developments for our application.

Keywords:

linear algebra, iOS application, Giac

Κατάλογος Εικόνων

Περιεχόμενα

Εισαγωγή

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η παρουσίαση της iOS εφαρμογής “Linear Algebra”. 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. Η εφαρμογή υλοποιεί λειτουργίες της γραμμικής άλγεβρας που αφορούν διανύσματα, πίνακες και επίλυση συστήματος εξισώσεων, καλώντας συναρτήσεις της Giac.

Στο πρώτο κεφάλαιο της εργασίας αναφέρουμε εφαρμογές με παρόμοιο αντικείμενο με αυτό της εφαρμογής μας. Αρχικά παρουσιάζουμε τις πιο δημοφιλείς εφαρμογές για iOS και αναφέρουμε συνοπτικά πληροφορίες για αυτές, ενώ τονίζουμε σε τι διαφέρουν από τη δική μας, παραθέτοντας, στη συνέχεια, τίτλους εφαρμογών για το λειτουργικό Android.

Στο δεύτερο κεφάλαιο παρουσιάζουμε το υπόβαθρο που πρέπει να διαθέτει κανείς για να μπορεί να ασχοληθεί με την δημιουργία εφαρμογών για iOS. Ενδεικτικά, αναφέρουμε τις γνώσεις προγραμματισμού και την εξοικείωση με την αντικειμενοστρέφεια, καθώς και με τα εργαλεία ανάπτυξης τέτοιων εφαρμογών.

Το τρίτο κεφάλαιο αφορά 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 σημαντικότερων εκ των εργαλείων που προσφέρει. Ακολούθως, εξετάζεται η C++ βιβλιοθήκη Giac, που αποτελεί τον υπολογιστικό πυρήνα της εφαρμογής και η οποία προσφέρει συναρτήσεις για τις λειτουργίες της γραμμικής άλγεβρας που μπορεί να εκτελέσει ο χρήστης.

Στο τέταρτο κεφάλαιο γίνεται αναλυτική παρουσίαση των προσφερόμενων λειτουργιών και της διεπαφής της εφαρμογής.

Στο πέμπτο κεφάλαιο κάνουμε μια μικρή αναφορά στα οφέλη που αποκομίσαμε δουλεύοντας στην παρούσα εργασία και παραθέτουμε πώς μπορεί να επεκταθεί η εφαρμογή στο μέλλον και να βελτιωθεί.

Στο τέλος της εργασίας μπορεί να βρεθεί παράρτημα όπου παρατίθεται ενδεικτικός κώδικας της εφαρμογής.

Κεφάλαιο 1ο

Σχετικές Εφαρμογές

Εφαρμογές σε iOS

Κατά τη διάρκεια εκπόνησης της παρούσας εργασίας, η εταιρεία "Wolfram Research", ιδρυτής και πρόεδρος της οποίας είναι ο Stephen Wolfram (δημιουργός του Mathematica), κυκλοφόρησε μια παρόμοια εφαρμογή με τη δικιά μας. Η εφαρμογή ονομάζεται "Linear Algebra Course Assistant" και τρέχει σε iPhone, iPod Touch και iPad (εικόνα 1). Σχετίζεται με πράξεις γραμμικής άλγεβρας, όπως αριθμητική πινάκων, ιδιοτιμές και ιδιοδιανύσματα, κ.ά., όπως και η δική μας εφαρμογή, με τη διαφορά ότι εμείς χρησιμοποιούμε τη βιβλιοθήκη Giac και τις συναρτήσεις της. Να σημειώσουμε ότι η ίδια εταιρεία έχει βγάλει βοηθητικές εφαρμογές και για άλλα μαθήματα, όπως στατιστική, άλγεβρα, φυσική και μηχανική.

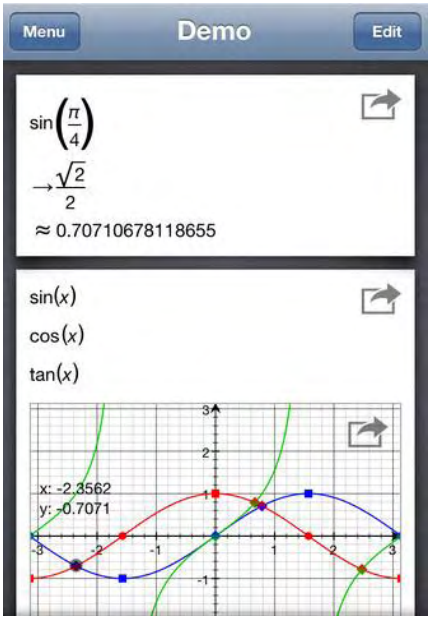
α) Αρχική οθόνη

β) Εισαγωγή διανυσμάτων για υπολογισμό εξωτερικού γινομένου

Εικόνα 1: Η εφαρμογή "Linear Algebra Course Assistant"

Μια άλλη εφαρμογή που χρησιμοποιεί και αυτή τη βιβλιοθήκη Giac είναι η "PocketCAS" (εικόνα 2), της εταιρείας "Eurocomp". Η εφαρμογή αυτή είναι μια υπολογιστική μηχανή και μία από τις δημοφιλέστερες και πληρέστερες εφαρμογές αυτού του είδους. Η ευρεία γκάμα των λειτουργιών που προσφέρει, όμως, την καθιστά αρκετά δύσχρηστη και απαιτητική σε

υπολογιστικούς πόρους. Η εφαρμογή μας αποτελεί ένα υποσύνολο αυτής, καθώς εξειδικεύεται σε πράξεις γραμμικής άλγεβρας, προσφέροντας μια πιο φιλική διεπαφή στο χρήστη.



α) Υπολογισμός ημιτόνου

β) Εισαγωγή σειράς taylor

Εικόνα 2: Η εφαρμογή "PocketCAS"

Τέλος, άλλες δύο εφαρμογές που θα μπορούσαν να αναφερθούν είναι η "HiCalc PRO" (εικόνα 3) της PPCLINK Mobile Software και η "Calc Pro" (εικόνα 4) της Panoramic Software Inc. Αυτές είναι κατά κύριο λόγο εφαρμογές για υπολογιστικές πράξεις, με κάποιες επιπρόσθετες δυνατότητες, όπως επίλυση εξισώσεων.

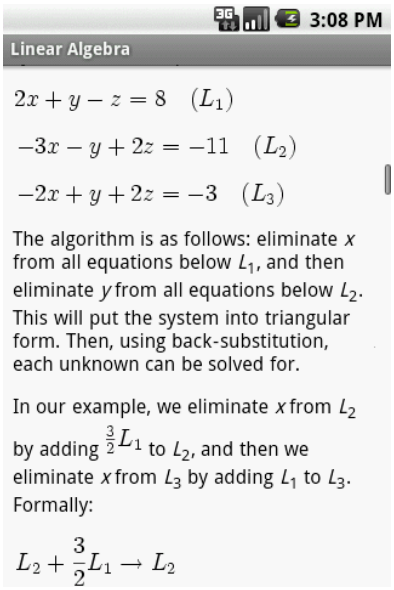
Εικόνα 3: Η εφαρμογή "HiCalc PRO"

Εικόνα 4: Η εφαρμογή "Calc Pro"

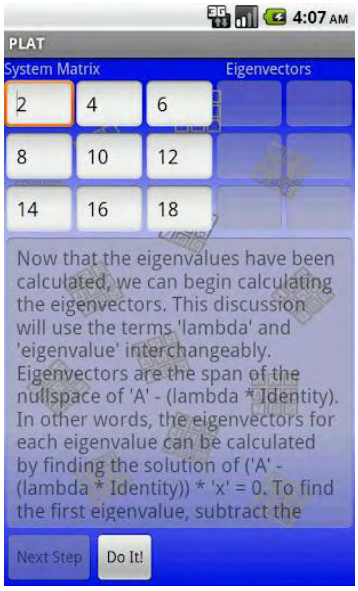
Εφαρμογές σε λειτουργικό Android

Στο λειτουργικό Android υπάρχει μεγάλη ποικιλία εφαρμογών και πολλές κλασικές αριθμομηχανές (calculators), όμως εφαρμογές που εξειδικεύονται στη γραμμική άλγεβρα, όπως η "PocketCAS" και η " Linear Algebra Course Assistant" που αναφέραμε στην προηγούμενη ενότητα δεν υπάρχουν.

Μπορούμε να αναφέρουμε ότι υπάρχουν εφαρμογές – βοηθήματα για κατανόηση εννοιών της γραμμικής άλγεβρας, όπως τους πίνακες, τα ιδιοδιανύσματα και ορθοκανονικά διανύσματα για παράδειγμα. Τέτοιου είδους είναι η "Linear Algebra Study Guide" (εικόνα 5) και η "Pocket Linear Algebra Tutor" (εικόνα 6), οι οποίες, μάλιστα, δεν διατίθενται δωρεάν στους χρήστες.



Εικόνα 5: Η εφαρμογή "Linear Algebra Study Guide"



Εικόνα 6: Η εφαρμογή "Pocket Linear Algebra Tutor"

Η μόνη εφαρμογή σε λειτουργικό Android που καλύπτει μεγάλο μέρος του αντικειμένου της γραμμικής άλγεβρας, αλλά και των λειτουργιών της εφαρμογής μας, είναι η "Matrix Calculator" (εικόνα 7) του χρήστη Sangsin An, η οποία έχει μια έκδοση που διατίθεται δωρεάν και μία στην τιμή των 3,88 Euro, με την πρώτη να έχει σημειώσει 1000 – 5000 μεταφορτώσεις στο διάστημα από 19 Μαΐου έως 19 Ιουνίου 2012. Με την εφαρμογή αυτή, εκτός των συνηθισμένων πράξεων όπως εύρεση αντιστρόφου, ορίζουσας και ιδιοτιμών, μπορεί ο χρήστης να κάνει παραγοντοποιήσεις (Cholesky, LU, QR) και επίλυση γραμμικών εξισώσεων. Να σημειώσουμε ότι όλα τα παραπάνω είναι χαρακτηριστικά και της εφαρμογής που υλοποιούμε σε αυτή την εργασία.

Εικόνα 7: Η εφαρμογή "Matrix Calculator"

Κεφάλαιο 2ο

Θεωρητικό Υπόβαθρο

Γνώσεις προγραμματισμού

Για να υλοποιήσει κάποιος μία εφαρμογή σαν αυτή της παρούσας εργασίας, είναι απαραίτητο να είναι εξοικειωμένος με όρους αντικειμενοστρέφειας, όπως κλάση, αντικείμενο, μεθόδους, κ.ά. και να ξέρει κάποια γλώσσα προγραμματισμού που είναι βασισμένη στη C (C, C++, C#, Objective – C). Το πρόγραμμα σπουδών της σχολής μας μας δίνει την ευκαιρία να αποκτήσουμε τα παραπάνω εφόδια, καθώς διδασκόμαστε τη γλώσσα C στο πρώτο εξάμηνο και την αντικειμενοστραφή γλώσσα Java στο δεύτερο εξάμηνο του πρώτου έτους. Επίσης, πολλά μαθήματα επόμενων ετών απαιτούν την περαιτέρω ενασχόληση με αυτές τις γλώσσες προγραμματισμού, γεγονός που προσφέρει το προγραμματιστικό υπόβαθρο για να ασχοληθεί κανείς με την ανάπτυξη εφαρμογών σε iOS.

Εξοικείωση με την Objective – C και το Xcode

Η Objective – C είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού, παρακλάδι της C γλώσσας, και μας επιτρέπει να ορίζουμε κλάσεις και αντικείμενα. Αν κάποιος διαθέτει γνώσεις αντικειμενοστραφούς προγραμματισμού εύκολα μπορεί να εξοικειωθεί με τη γλώσσα, αρκεί να προσέξει μια σημαντική της διαφορά: η Objective – C προσφέρει μηχανισμούς για διαχείριση της μνήμης.

Το Xcode είναι το περιβάλλον ανάπτυξης εφαρμογών για το iOS. Κάποιος που διαθέτει υπολογιστή της Apple Inc. είναι πολύ πιθανό να το γνωρίζει και να το έχει χρησιμοποιήσει. Θα αναφερθούμε αναλυτικότερα σε αυτό και στην Objective – C σε επόμενη ενότητα.

Προσωπικά, δεν έχω Mac υπολογιστή, δεν είχα ασχοληθεί με αυτή τη γλώσσα και πιθανόν να μου ήταν δύσκολο να τη μάθω μόνος μου για να υλοποιήσω εφαρμογές που να τρέχουν σε iOS συσκευές. Η ευκαιρία προέκυψε όταν δημιουργήθηκε στη σχολή μας ένα καινούριο μάθημα σε πλαίσιο ειδικής εργασίας, το οποίο θα είχε ως αντικείμενο την ανάπτυξη εφαρμογών σε iOS. Επιλέγοντας αυτό το μάθημα, εγώ και άλλοι συμφοιτητές μου, ήρθαμε για πρώτη φορά σε επαφή με την Objective – C και τα προγραμματιστικά εργαλεία του iOS και κατασκευάσαμε την πρώτη μας εφαρμογή για iPhone, ακολουθώντας διαλέξεις του πανεπιστημίου του Stanford. Έτσι, απέκτησα εμπειρία που θα μου επέτρεπε να επιλέξω την παρούσα διπλωματική εργασία.

Κεφάλαιο 3ο

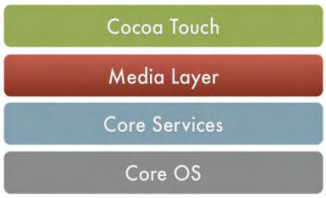
Περιβάλλον Εργασίας

Το λειτουργικό σύστημα iOS

Το iOS είναι το Unix-like λειτουργικό σύστημα για τις φορητές συσκευές της Apple Inc. Τη στιγμή της συγγραφής αυτής της εργασίας (Ιούνιος 2012), οι iOS εφαρμογές στο App Store – την πλατφόρμα συγκέντρωσης και διαμοίρασης όλων των iOS εφαρμογών – ξεπερνούν σε αριθμό τις 650.000, οι οποίες συνολικά έχουν μεταφορτωθεί πάνω από 30 δισεκατομμύρια φορές. Η τελευταία έκδοση του iOS (ως τον Ιούνιο του 2012) είναι το iOS 5.1.1., ενώ στο WWDC (Worldwide Developers Conference) στις 11 Ιουνίου ανα-κοινώθηκε και το iOS 6.

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 8).



Εικόνα 8: iOS abstraction layers

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 και iPad (εικόνα 9). 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, Wi-Fi, Bluetooth.

Η αναβάθμιση του υπάρχοντος λειτουργικού συστήματος είναι μια εύκολη διαδικασία που πραγματοποιείται μέσα από το 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 9: Συσκευές iOS

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 και iPad Simulator). Το περιβάλλον ανάπτυξης λογισμικού iOS είναι το Xcode, το οποίο θα εξετάσουμε αναλυτικά στην επόμενη ενότητα. Η τελευταία έκδοση (έως τον Ιούνιο 2012) είναι το iOS 6 SDK (beta), ενώ η τελευταία του Xcode είναι το Xcode 4.5.

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 προ-γραμματισμού.

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

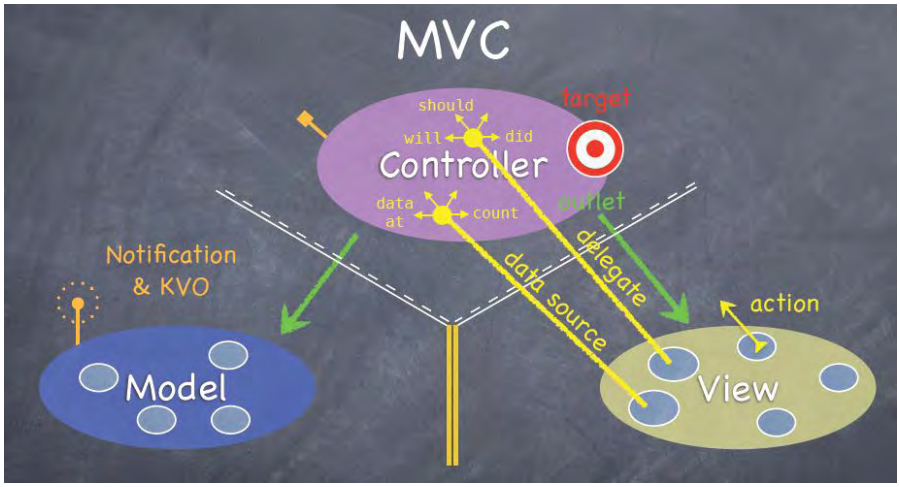
Τα "properties" είναι χαρακτηριστικό της αναθεωρημένης έκδοσης της Objective – C με το όνομα Objective – C 2.0 και ουσιαστικά αποτελούν συντακτική διευκόλυνση για τη δήλωση και διαχείριση instance variables (μεταβλητών στιγμιότυπου).

Reference Counting

Επίσης χαρακτηριστικό της Objective – C 2.0 (και του iOS συγκεκριμένα), το "reference counting" είναι ένας μηχανισμός διαχείρισης μνήμης πού 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 τον 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 (εικόνα 10). 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.



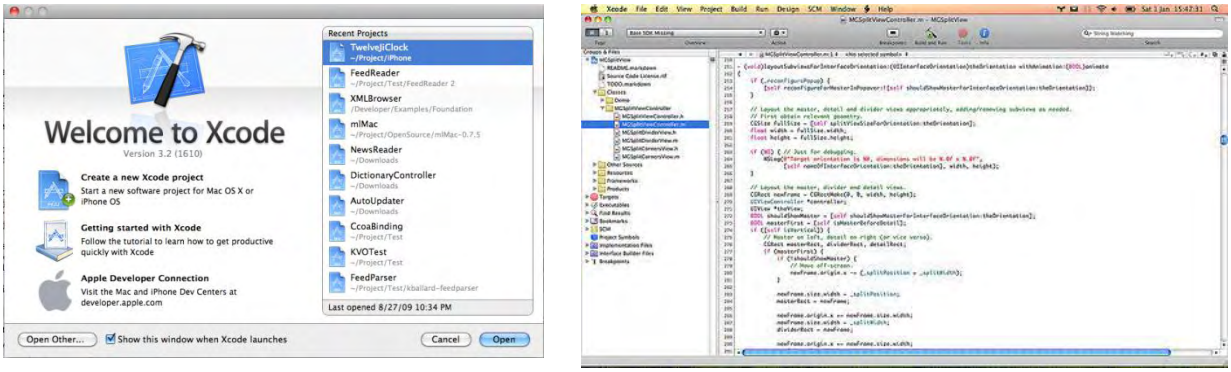
Εικόνα 10: Το MVC μοντέλο σχεδίασης

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. 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 τούς στο App Store.

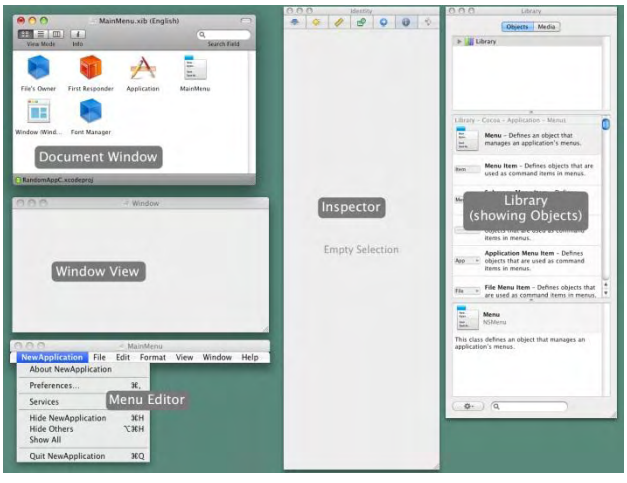


α) Η Αρχική οθόνη β) Κυρίως περιβάλλον

Εικόνα 11: Xcode 3

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 ή 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 Interface 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.



Εικόνα 12: Ο Interface Builder του Xcode 3

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.

Εικόνα 13: iOS simulator

Η βιβλιοθήκη Giac

Η Giac είναι μια δωρεάν μαθηματική βιβλιοθήκη γραμμένη σε C++, 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.

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 περισσό-

τερες από αυτές είναι βοηθητικές βιβλιοθήκες (όπως οι ‘libconv’, ‘ncurses’, ‘gettext’ και ‘readline’) που προσφέρουν περιφερειακές λειτουργίες, το πιο σημαντικό σημείο όμως είναι η ύπαρξη μιας βιβλιοθήκης αριθμητικής μεγάλης ακριβείας, είτε της GMP (Gnu Multiple Precision 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 είναι η 0.9.8 (stable version), αλλά για την εφαρμογή χρησιμοποιήθηκε η έκδοση 0.9.5.

Πέρα από τις προαπαιτούμενες αυτές βιβλιοθήκες, η Giac έχει τη δυνατότητα αξιοποίησης επιπλέον μαθηματικών βιβλιοθηκών, οι οποίες είναι προαιρετικές. Αυτές είναι οι MPFR, NTE, 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.

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/0 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/ δια-νύσματα, 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:
   a) μπορεί να ελεγχθεί ως εξής: e.type==_INT_
   b) λήψη τιμής γράφοντας: int i=e.val;
2. Αριθμός κινητής υποδιαστολής διπλής ακριβείας double:
   a) μπορεί να ελεγχθεί ως εξής: e.type==_DOUBLE_
   b) λήψη τιμής γράφοντας: double d=e._DOUBLE_val;
3. Ακέραιος μεγάλης ακριβείας:
   a) μπορεί να ελεγχθεί ως εξής: e.type==_ZINT
   b) 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:
   a) μπορεί να ελεγχθεί ως εξής: .type==_REAL
5. Μιγαδικός αριθμός:
   a) μπορεί να ελεγχθεί ως εξής: e.type==_CINT
   b) ο αντίστοιχος δείχτης 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):
   a) μπορεί να ελεγχθεί ως εξής: e.type==_VECT
   b) 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:
   a) μπορεί να ελεγχθεί ως εξής: e.type==_IDNT
   b) 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:
   a) μπορεί να ελεγχθεί ως εξής: e.type==_SYMB
   b) ο αντίστοιχος δείκτης δείχνει στον τύπο symbolic: symbolic s=*e._SYMBptr; Περισσότερες λεπτομέρειες μπορούν να βρεθούν στο αρχείο giac/symbolic.h.
9. Αλφαριθμητικό (string):
   a) μπορεί να ελεγχθεί ως εξής: e.type==_STRING με αντίστοιχο δείκτη *e.type==_STRING
10. Αντικείμενο – συνάρτηση:
    a) μπορεί να ελεγχθεί ως εξής: e.type==_FUNC
    b) 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.

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 "test" τον πίνακα "[[2,3],[1,0]]", ένας τρόπος θα ήταν ο εξής:

```
gen test("[[2,3],[1,0]]",0);
ή
gen test = gen("[[2,3],[1,0]]",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'.

Συναρτήσεις στη Giac

Στη βιβλιοθήκη της Giac, κάθε συνάρτηση θεωρείται ότι δέχεται μία παράμετρο και επιστρέφει μια άλλη, χρησιμοποιώντας τον τύπο 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() πού 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.

Αν ψάχνουμε για μία συνάρτηση για να εκτελέσουμε κάποιον υπολογισμό, μπορούμε να απευθυνθούμε στο documentation του Xcas, μιας και Giac και Xcas χρησιμοποιούν τις ίδιες συναρτήσει με ελαφρώς παραλλαγμένα ονόματα. Αν το όνομα αυτής της συνάρτησης υπάρχει, μπορούμε να την καλέσουμε προσθέτοντας, συνήθως, το πρόθεμα «_» στην αρχή του ονόματος, δηλαδή:

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

όπου το δεύτερο όρισμα θέτουμε πάντα '0' στην εφαρμογή μας.

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 πού 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 τούς ακεραίους 4 και 12:

  ```
  gen a = gen(4,0);        // type : int
  gen b = gen(12,0);       // 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 επιστρεφόμενο αποτέλεσμα σε μια μεταβλητή τύπου gen με το όνομα result:

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

Κεφάλαιο 4ο

Παρουσίαση της εφαρμογής

Προσφερόμενες Λειτουργίες

Η εφαρμογή δίνει τη δυνατότητα στο χρήστη να εκτελέσει λειτουργίες της Γραμμικής Άλγεβρας, που αφορούν διανύσματα και πίνακες. Οι λειτουργίες αυτές χωρίζονται σε πέντε κατηγορίες:

1. Πράξεις που αφορούν διανύσματα
2. Πράξεις που αφορούν πίνακες
3. Παραγοντοποιήσεις πινάκων
4. Επίλυση γραμμικών εξισώσεων

Στη συνέχεια, περιγράφουμε αναλυτικότερα τις λειτουργίες κάθε κατηγορίας, παραθέτοντας και το όνομα της συνάρτησης της Giac που χρησιμοποιήσαμε.

Σε ό,τι αφορά την πρώτη κατηγορία ο χρήστης μπορεί να υπολογίσει:

- Την απόλυτη τιμή ενός διανύσματος (_abs)
- Την ευκλείδεια (δεύτερη) νόρμα ενός διανύσματος (_l2norm)
- Τη νόρμα με το μέγιστο ενός διανύσματος (ή ενός πίνακα) (_maxnorm)
- Το εξωτερικό γινόμενο δύο διανυσμάτων (_cross)
- Το εσωτερικό γινόμενο δύο διανυσμάτων (_dotprod)
- Το άθροισμα δύο διανυσμάτων (+)
- Το αποτέλεσμα αφαίρεσης δύο διανυσμάτων (-)

Στις λειτουργίες της δεύτερης κατηγορίας ο χρήστης μπορεί να υπολογίσει:

- Τον ανάστροφο ενός πίνακα (_tran)
- Τον αντίστροφο ενός πίνακα (_inv)
- Την ορίζουσα ενός πίνακα (_det)
- Τον πίνακα που προκύπτει βάζοντας 0 σε μια στήλη, σύμφωνα με τη μέθοδο Gauss – Jordan με συγκεκριμένο στοιχείο ως οδηγό (_pivot)
- Την τάξη ενός πίνακα (_rank)
- Τις ιδιοτιμές ενός πίνακα (_eigenvals)
- Τα ιδιοδιανύσματα ενός πίνακα (_egv)
- Το συζυγή ανάστροφο ενός πίνακα (_trn)
- Τις λύσεις ενός γραμμικού συστήματος εξισώσεων με την αναγωγή Gauss – Jordan (_rref)
- Τη λύση ενός προβλήματος γραμμικού προγραμματισμού εύρεσης μέγιστου (ή ελάχιστου) μιας συνάρτησης χρησιμοποιώντας τη μέθοδο Simplex (_simplex_reduce)

- Τον πυρήνα ενός πίνακα (_ker)
- Την εικόνα ενός πίνακα (_image)
- Το ίχνος ενός πίνακα (_ckmtrace)
- Το χαρακτηριστικό πολυώνυμο ενός πίνακα (_pcar)
- Το γινόμενο Hadamard δύο πινάκων (_hadamard)
- Το άθροισμα δύο πινάκων (+)
- Το αποτέλεσμα αφαίρεσης δύο πινάκων (-)
- Το γινόμενο δύο πινάκων (*)

Στην τρίτη κατηγορία λειτουργιών δίνεται η δυνατότητα να παραγοντοποιηθεί ένας πίνακας με τις εξής μεθόδους:

- LU (lu)
- Cholesky (_cholesky)
- QR (qr)
- SVD (_svd)
- LLL (_lll)

Τέλος, στην τελευταία κατηγορία ο χρήστης μπορεί να υπολογίσει:

- Τη βάση ενός χώρο παραγόμενο από ένα σύνολο διανυσμάτων (_basis)
- Τη λύση ενός συστήματος γραμμικών εξισώσεων (_linsolve)

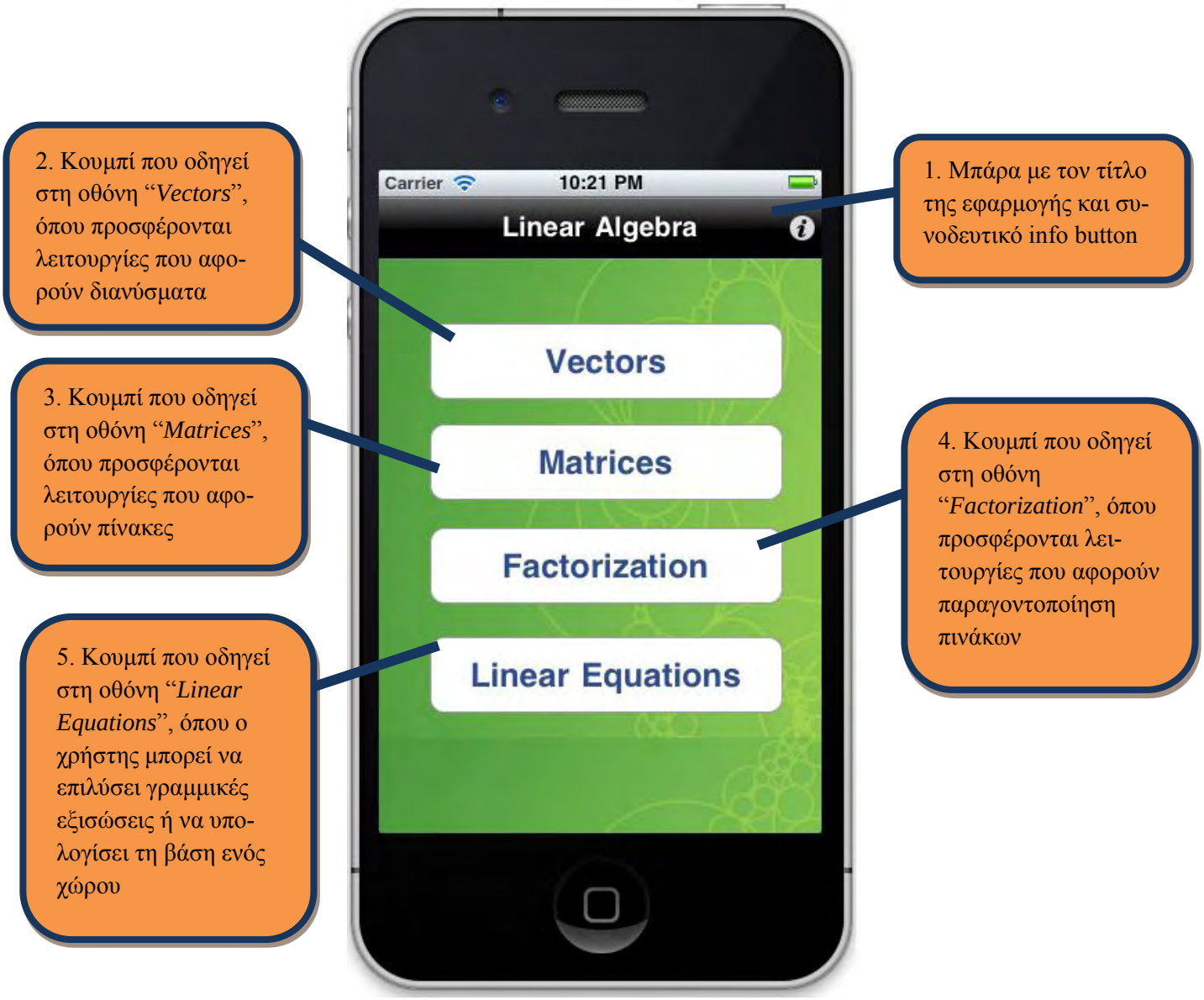
Διεπαφή της εφαρμογής (Interface)

Τα κύρια χαρακτηριστικά που επιδιώξαμε για τη διεπαφή (Interface) της εφαρμογής μας είναι η απλότητα, η χρηστικότητα και η καλαισθησία. Πολλές εφαρμογές που κυκλοφορούν στην αγορά, μέρος των οποίων περιγράψαμε στην πρώτη ενότητα, ενσωματώνουν πολλές λειτουργίες σε μία μόνο εφαρμογή, καθιστώντας την δύσκολη στη χρήση σε μια φορητή συσκευή περιορισμένου μεγέθους οθόνης, όπως το iPhone. Για αυτό το λόγο, προσπαθήσαμε να υλοποιήσουμε συγκεκριμένες λειτουργίες που αφορούν τον τομέα της Γραμμικής Άλγεβρας και να τις οργανώσουμε κατάλληλα, ώστε να διευκολύνουν τη χρήση τους σε ένα κινητό τηλέφωνο.

Στην αρχική οθόνη της εφαρμογής υπάρχουν τα εξής αντικείμενα (συγκεντρωτικά εμφανίζονται στην εικόνα 14):

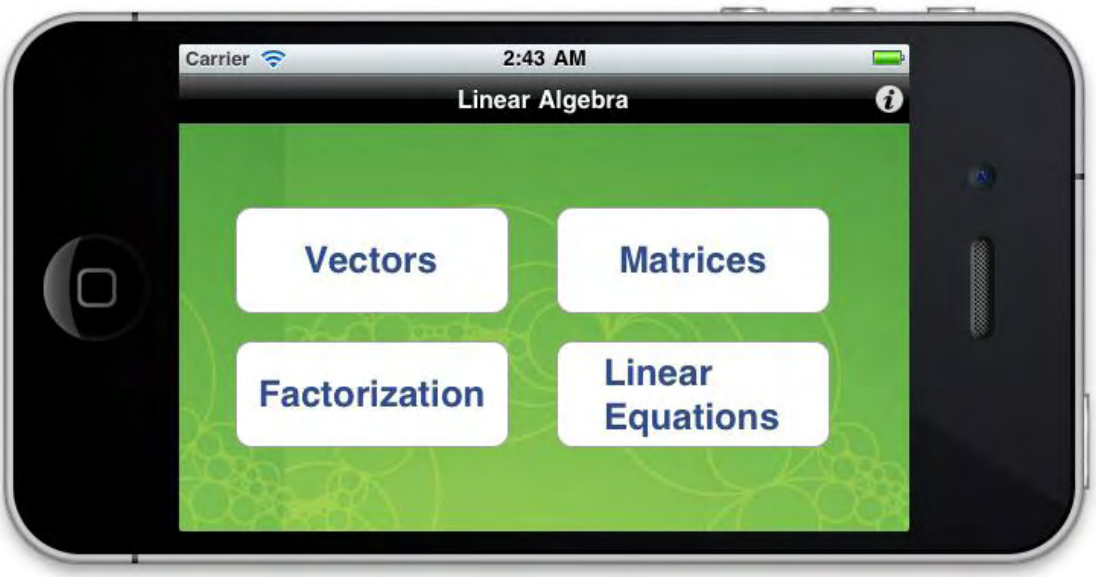
1. Μια μπάρα με τον τίτλο της εφαρμογής στο πάνω μέρος της οθόνης, συνοδευόμενη από ένα info button που εμφανίζει πληροφορίες για αυτή σε ένα alert window αν πατηθεί.
2. Κουμπί με τίτλο “Vectors” που, αν πατηθεί, οδηγεί στην οθόνη “Vectors”, όπου προσφέρονται λειτουργίες που αφορούν πράξεις με ένα διάνυσμα και περιγράφονται στην πρώτη κατηγορία της ενότητας 4.1.

3. Κουμπί με τίτλο “Matrices” που, αν πατηθεί, οδηγεί στην οθόνη “Matrices”, όπου προσφέρονται λειτουργίες που αφορούν πίνακες και περιγράφονται στη δεύτερη κατηγορία της ενότητας 4.1.
4. Κουμπί με τίτλο “Factorization” που, αν πατηθεί, οδηγεί στην οθόνη “Factorization”, όπου προσφέρονται λειτουργίες που αφορούν παραγοντοποίηση πινάκων και περιγράφονται στην τρίτη κατηγορία της ενότητας 4.1.
5. Κουμπί με τίτλο “Linear Equations” που, αν πατηθεί, οδηγεί στην οθόνη “Linear Equations”, όπου ο χρήστης μπορεί να επιλύσει γραμμικές εξισώσεις ή να υπολογίσει τη βάση ενός χώρου (τελευταία κατηγορία της ενότητας 4.1).



Εικόνα 14: Η αρχική οθόνη της εφαρμογής μας

Αν ο χρήστης αλλάξει τον προσανατολισμό της συσκευής του, αλλάζει και ο προσανατολισμός της οθόνης της εφαρμογής μας. Έτσι, η οθόνη περιστρέφεται και παίρνει τη μορφή της εικόνας 15. Επίσης, στην εικόνα 16 απεικονίζεται το alert window που εμφανίζεται όταν πατήσουμε το info button δεξιά του τίτλου της εφαρμογής μας.

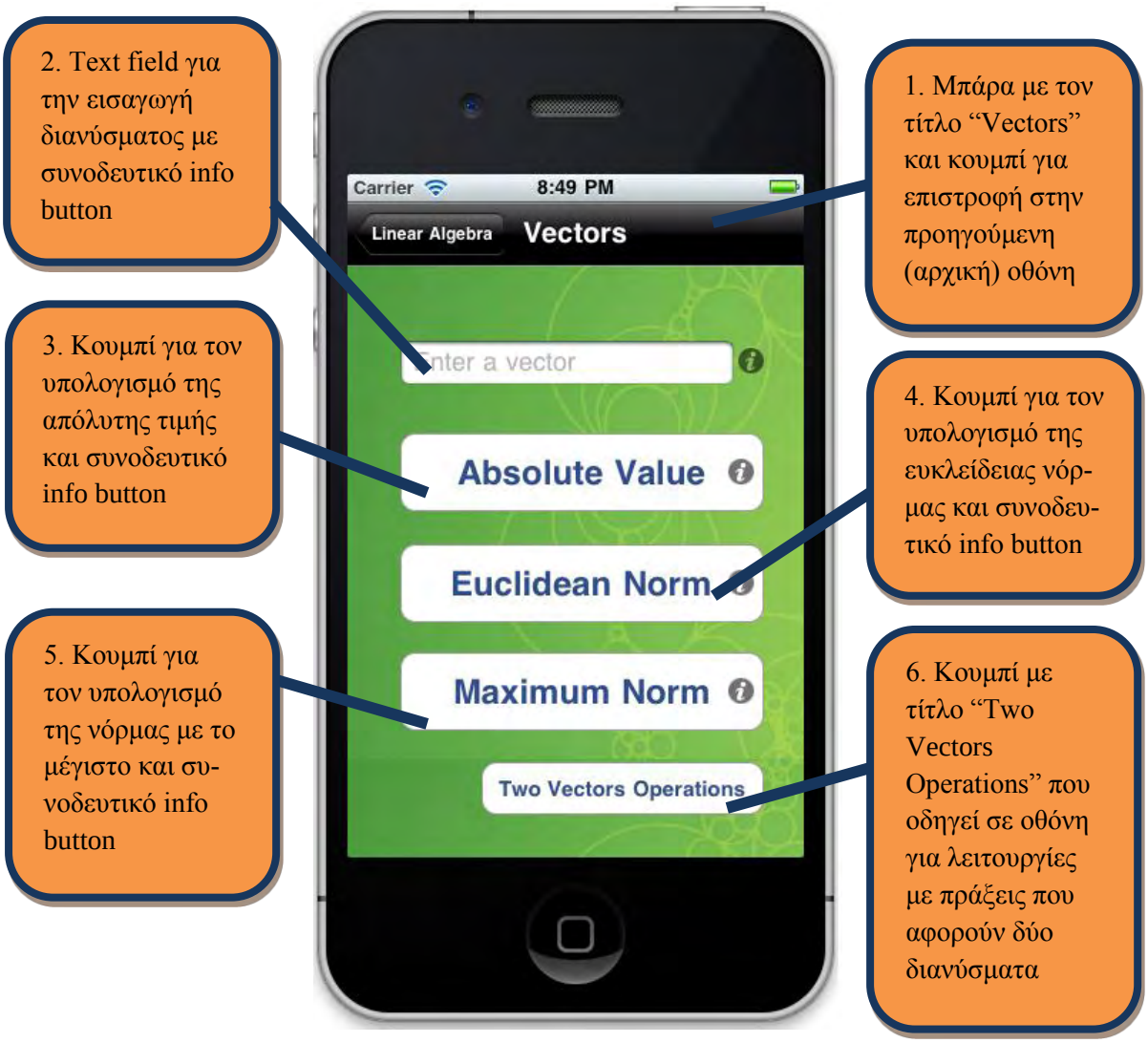


Εικόνα 15: Η αρχική οθόνη της εφαρμογής μας όταν ο χρήστης περιστρέψει το κινητό του

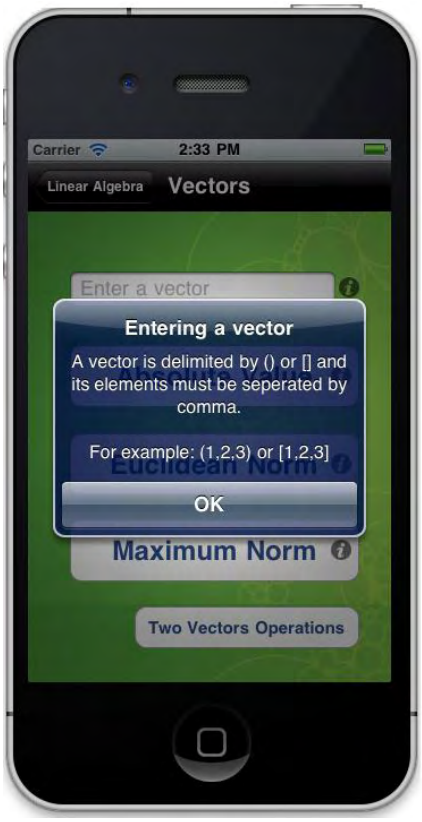
Εικόνα 16: UIAlertView με κάποιες πληροφορίες για την εφαρμογή μας

Πατώντας το κουμπί “Vectors”, ο χρήστης οδηγείται στην οθόνη με τον ομώνυμο τίτλο, στην οποία υπάρχουν τα εξής αντικείμενα (συγκεντρωτικά εμφανίζονται στην εικόνα 17):

1. Μια μπάρα με τον τίτλο “Vectors” στο πάνω μέρος της οθόνης και ένα κουμπί για την επιστροφή στην προηγούμενη (αρχική) οθόνη.
2. Text field με προκαθορισμένο κείμενο “Enter a vector” και συνοδευτικό info button με οδηγίες για τη σωστή εισαγωγή διανύσματος. Όταν ο χρήστης ακουμπήσει πάνω του, εμφανίζεται πληκτρολόγιο για να εισάγει το διάνυσμα.
3. Κουμπί για τον υπολογισμό της απόλυτης τιμής του δοθέντος διανύσματος, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
4. Κουμπί για τον υπολογισμό της ευκλείδειας (δεύτερης) νόρμας του δοθέντος διανύσματος, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
5. Κουμπί για τον υπολογισμό της νόρμας με το μέγιστο του δοθέντος διανύσματος, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
6. Κουμπί με τίτλο “Two Vectors Operations” που, αν πατηθεί, οδηγεί στη δεύτερη οθόνη με τίτλο “Vectors”, όπου προσφέρονται λειτουργίες που αφορούν πράξεις με δύο διανύσματα και περιγράφονται στην πρώτη κατηγορία της ενότητας 4.1.



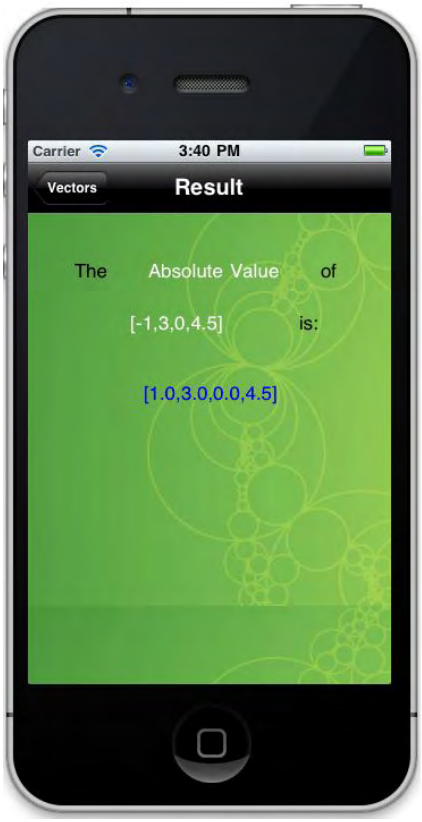
Εικόνα 17: Η οθόνη “Vectors” με πράξεις που αφορούν ένα διάνυσμα



Εικόνα 19: Εμφάνιση alert window για την εισαγωγή διανύσματος

Εικόνα 18: Εισαγωγή διανύσματος μέσα από το πληκτρολόγιο

Εικόνα 21: Εμφάνιση alert window με πληροφορίες



Εικόνα 20: Η οθόνη με το αποτέλεσμα

Αν ο χρήστης βρίσκεται στη οθόνη με τίτλο “Vectors” της εικόνας και πατήσει το κουμπί “Two Vectors Operations”, θα βρεθεί στη δεύτερη οθόνη με τίτλο “Vectors”, η οποία όμως του επιτρέπει να εκτελέσει πράξεις με δύο διανύσματα. Σε αυτή την οθόνη υπάρχουν τα εξής αντικείμενα (συγκεντρωτικά εμφανίζονται στην εικόνα 22):

1. Μια μπάρα με τον τίτλο “Vectors” στο πάνω μέρος της οθόνης και ένα κουμπί για την επιστροφή στην προηγούμενη οθόνη.
2. Text field με προκαθορισμένο κείμενο “Enter a vector”. Όταν ο χρήστης ακουμπήσει πάνω του, εμφανίζεται πληκτρολόγιο για να εισάγει το διάνυσμα.
3. Text field με προκαθορισμένο κείμενο “Enter a vector ”. Όταν ο χρήστης ακουμπήσει πάνω του, εμφανίζεται πληκτρολόγιο για να εισάγει το διάνυσμα.
4. Κουμπί για τον υπολογισμό του εξωτερικού γινομένου των δοθέντων διανυσμάτων, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
5. Κουμπί για τον υπολογισμό του εσωτερικού γινομένου των δοθέντων διανυσμάτων, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
6. Κουμπί για τον υπολογισμό του αθροίσματος των δοθέντων διανυσμάτων.
7. Κουμπί για τον υπολογισμό του αποτελέσματος της αφαίρεσης των δοθέντων διανυσμάτων.

Εικόνα 22: Η οθόνη “Vectors” για πράξεις με δύο διανύσματα

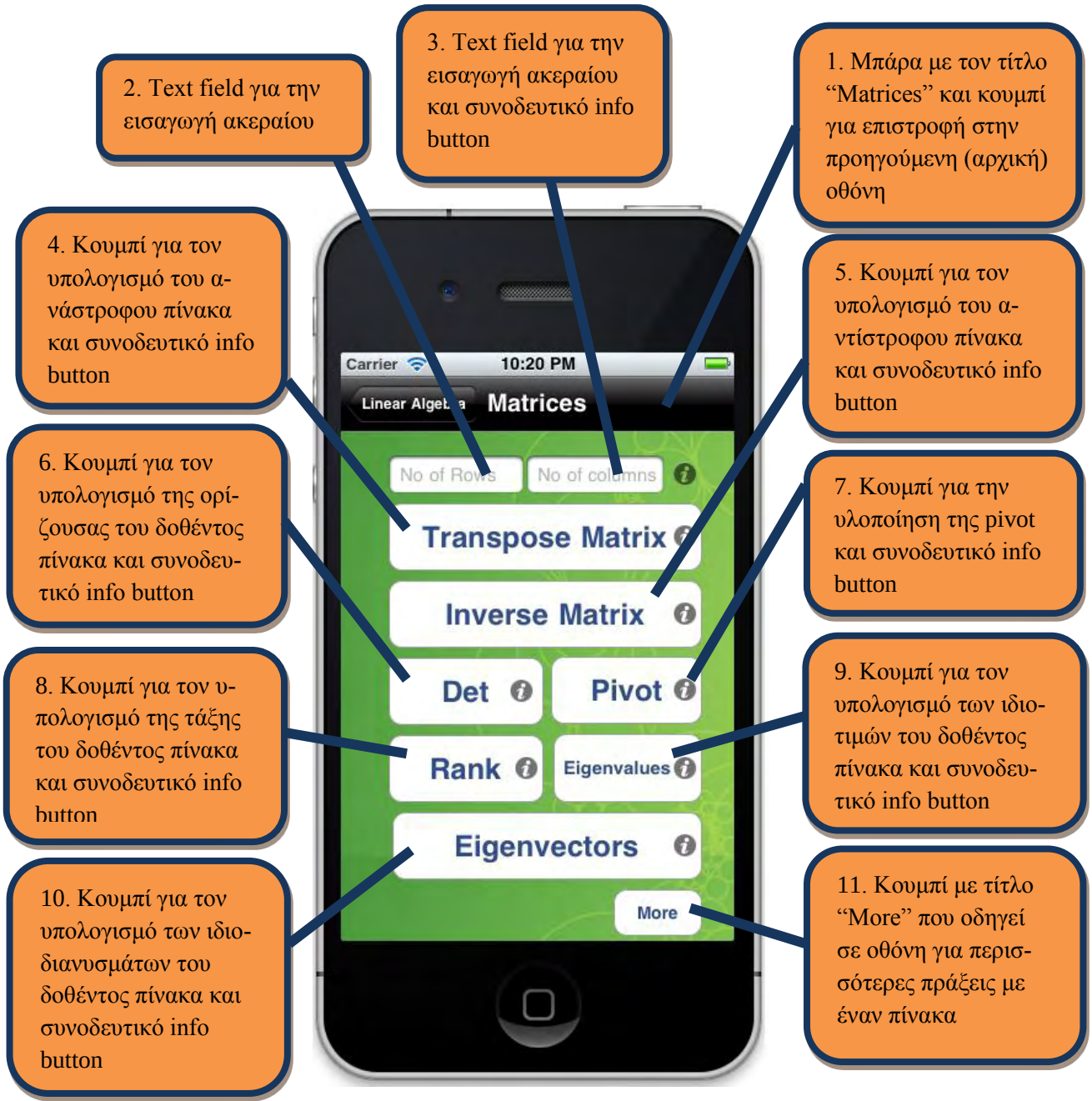
Αν από την αρχική οθόνη, αν ο χρήστης πατήσει το κουμπί με τίτλο “Matrices”, θα βρεθεί στην ομότιτλη οθόνη, η οποία του επιτρέπει να εκτελέσει πράξεις με έναν πίνακα. Σε αυτή την οθόνη υπάρχουν τα εξής αντικείμενα (συγκεντρωτικά εμφανίζονται στην εικόνα 23):

1. Μια μπάρα με τον τίτλο “Matrices” στο πάνω μέρος της οθόνης και ένα κουμπί για την επιστροφή στην προηγούμενη (αρχική) οθόνη.
2. Text field με προκαθορισμένο κείμενο “No. of rows”. Όταν ο χρήστης ακουμπήσει πάνω του, εμφανίζεται πληκτρολόγιο για να εισάγει έναν ακέραιο.
3. Text field με προκαθορισμένο κείμενο “No. of columns” συνοδευτικό info button. Όταν ο χρήστης ακουμπήσει πάνω του, εμφανίζεται πληκτρολόγιο για να εισάγει έ-ναν ακέραιο.
4. Κουμπί για τον υπολογισμό του ανάστροφου πίνακα, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
5. Κουμπί για τον υπολογισμό του αντίστροφου πίνακα, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
6. Κουμπί για τον υπολογισμό της ορίζουσας του δοθέντος πίνακα, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
7. Κουμπί για τον υπολογισμό του πίνακα που προκύπτει βάζοντας 0 σε μια στήλη, σύμφωνα με τη μέθοδο Gauss – Jordan με συγκεκριμένο στοιχείο ως οδηγό, συνο-δευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
8. Κουμπί για τον υπολογισμό της τάξης του δοθέντος πίνακα, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
9. Κουμπί για τον υπολογισμό των ιδιοτιμών του δοθέντος πίνακα, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
10. Κουμπί για τον υπολογισμό των ιδιοδιανυσμάτων του δοθέντος πίνακα, συνοδευόμε-νο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
11. Κουμπί με τίτλο “More” που, αν πατηθεί, οδηγεί στη δεύτερη οθόνη με τίτλο “Matrices”, όπου προσφέρονται περισσότερες λειτουργίες που αφορούν πράξεις με ένα πίνακα και περιγράφονται στη δεύτερη κατηγορία της ενότητας 4.1.

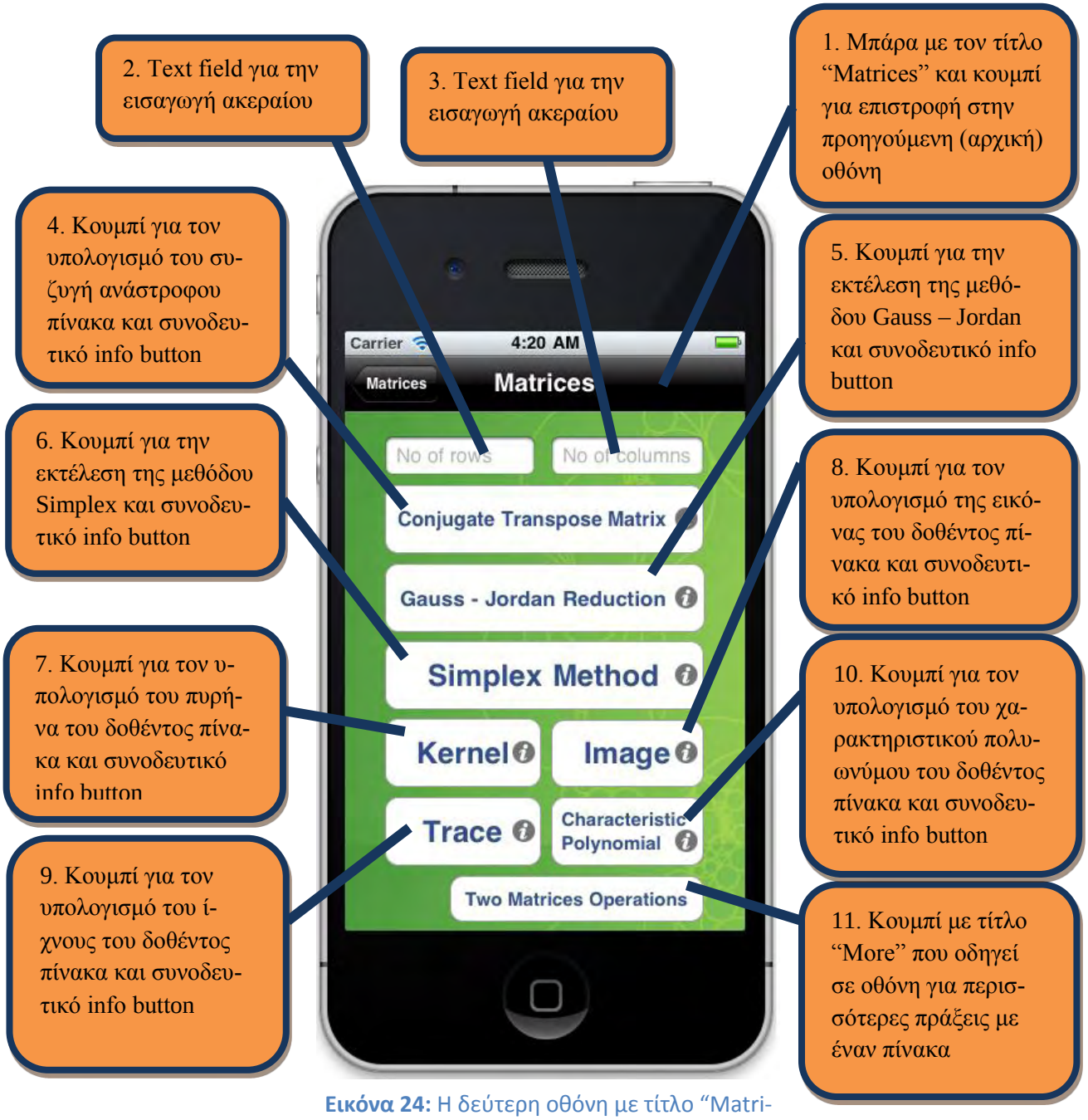
Από την οθόνη “Matrices” ο χρήστης μπορεί να βρεθεί στην οθόνη με περισσότερες λειτουρ-γίες που αφορούν ένα πίνακα πατώντας το κουμπί με τίτλο “More”. Σε αυτή την οθόνη υπάρ-χουν τα εξής αντικείμενα (συγκεντρωτικά εμφανίζονται στην εικόνα 24):

1. Μια μπάρα με τον τίτλο “Matrices” στο πάνω μέρος της οθόνης και ένα κουμπί για την επιστροφή στην προηγούμενη οθόνη.
2. Text field με προκαθορισμένο κείμενο “No. of rows”. Όταν ο χρήστης ακουμπήσει πάνω του, εμφανίζεται πληκτρολόγιο για να εισάγει έναν ακέραιο.
3. Text field με προκαθορισμένο κείμενο “No. of columns”. Όταν ο χρήστης ακουμπή-σει πάνω του, εμφανίζεται πληκτρολόγιο για να εισάγει έναν ακέραιο.
4. Κουμπί για τον υπολογισμό του συζυγή ανάστροφου πίνακα, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
5. Κουμπί για τον υπολογισμό των λύσεων ενός γραμμικού συστήματος εξισώσεων με την αναγωγή Gauss – Jordan, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
6. Κουμπί για την εκτέλεση της μεθόδου Simplex και συνοδευτικό info button.

7. Κουμπί για τον υπολογισμό του πυρήνα του δοθέντος πίνακα, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
8. Κουμπί για τον υπολογισμό της εικόνας του δοθέντος πίνακα, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
9. Κουμπί για τον υπολογισμό του ίχνους του δοθέντος πίνακα, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
10. Κουμπί για τον υπολογισμό του χαρακτηριστικού πολυωνύμου του δοθέντος πίνακα, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
11. Κουμπί με τίτλο "Two Matrices Operations" που, αν πατηθεί, οδηγεί στην τρίτη οθόνη με τίτλο "Matrices", όπου προσφέρονται λειτουργίες που αφορούν πράξεις με δύο πίνακες και περιγράφονται στη δεύτερη κατηγορία της ενότητας 4.1.



Εικόνα 23: Η οθόνη "Matrices" για πράξεις που αφορούν έναν πίνακα



Εικόνα 24: Η δεύτερη οθόνη με τίτλο "Matrices" για πράξεις που αφορούν έναν πίνακα

Από τη δεύτερη οθόνη "Matrices" που φαίνεται στην εικόνα και πατώντας το κουμπί με τίτλο "Two Matrices Operations", ο χρήστης βρίσκεται στην οθόνη που του επιτρέπει να εκτελέσει πράξεις με δύο πίνακες. Σε αυτή την οθόνη υπάρχουν τα εξής αντικείμενα (συγκεντρωτικά εμφανίζονται στην εικόνα 25):

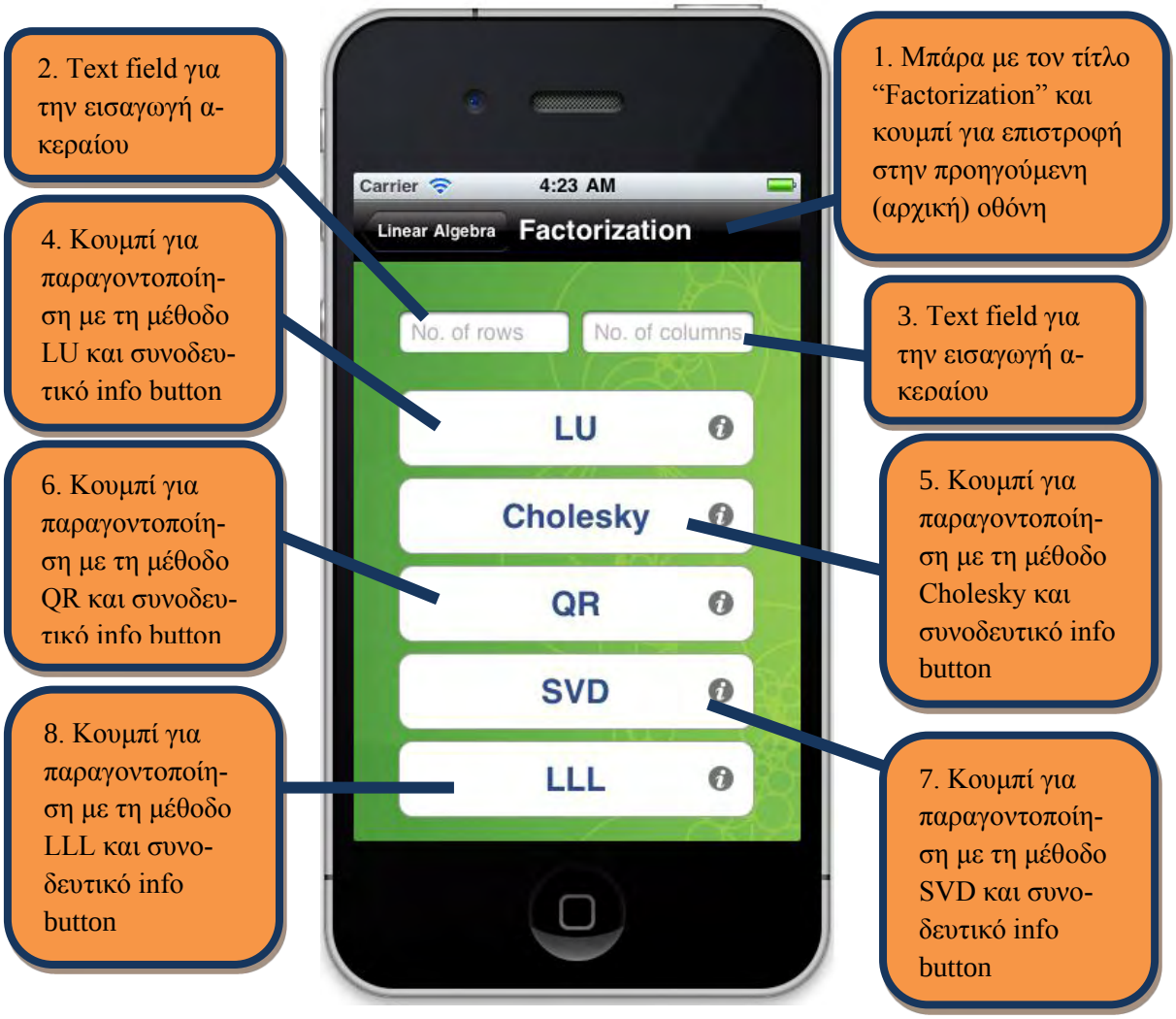
1. Μια μπάρα με τον τίτλο "Matrices" στο πάνω μέρος της οθόνης και ένα κουμπί για την επιστροφή στην προηγούμενη οθόνη.
2. Text field με προκαθορισμένο κείμενο "Enter a matrix". Όταν ο χρήστης ακουμπήσει πάνω του, εμφανίζεται πληκτρολόγιο για να εισάγει τον πίνακα.

3. Text field με προκαθορισμένο κείμενο “Enter a matrix”. Όταν ο χρήστης ακουμπήσει πάνω του, εμφανίζεται πληκτρολόγιο για να εισάγει τον πίνακα.
4. Κουμπί για τον υπολογισμό του γινομένου Hadamard, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
5. Κουμπί για τον υπολογισμό του αθροίσματος δύο πινάκων.
6. Κουμπί για τον υπολογισμό του αποτελέσματος αφαίρεσης δύο πινάκων.
7. Κουμπί για τον υπολογισμό του γινομένου δύο πινάκων.

Εικόνα 25: Η οθόνη με τίτλο “Matrices” για πράξεις που αφορούν δύο πίνακες

Αν από την αρχική οθόνη, αν ο χρήστης πατήσει το κουμπί με τίτλο “Factorization”, θα βρεθεί στην ομότιτλη οθόνη, η οποία του επιτρέπει να παραγοντοποιήσει έναν πίνακα, με τις μεθόδους που περιγράφονται στην τρίτη κατηγορία της ενότητας 4.1. Σε αυτή την οθόνη υπάρχουν τα εξής αντικείμενα (συγκεντρωτικά εμφανίζονται στην εικόνα 26):

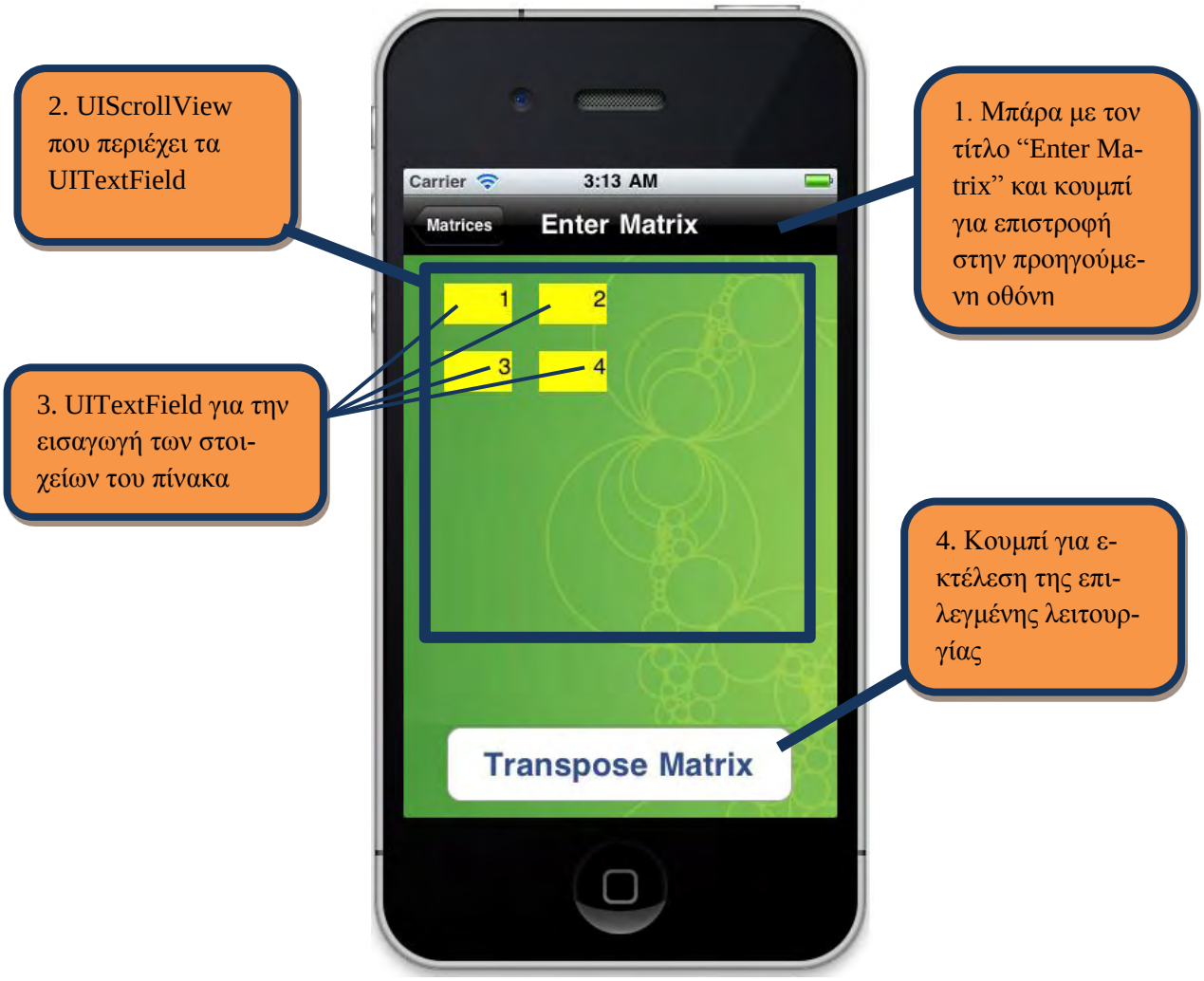
1. Μια μπάρα με τον τίτλο “Factorization” στο πάνω μέρος της οθόνης και ένα κουμπί για την επιστροφή στην προηγούμενη (αρχική) οθόνη.
2. Text field με προκαθορισμένο κείμενο “No. of rows”. Όταν ο χρήστης ακουμπήσει πάνω του, εμφανίζεται πληκτρολόγιο για να εισάγει έναν ακέραιο.
3. Text field με προκαθορισμένο κείμενο “No. of columns”. Όταν ο χρήστης ακουμπήσει πάνω του, εμφανίζεται πληκτρολόγιο για να εισάγει έναν ακέραιο.
4. Κουμπί για παραγοντοποίηση με τη μέθοδο LU, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
5. Κουμπί για παραγοντοποίηση με τη μέθοδο Cholesky, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
6. Κουμπί για παραγοντοποίηση με τη μέθοδο QR, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
7. Κουμπί για παραγοντοποίηση με τη μέθοδο SVD, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.
8. Κουμπί για παραγοντοποίηση με τη μέθοδο LLL, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.



Εικόνα 26: Η οθόνη “Factorization” για παραγοντοποίηση ενός πίνακα

Όπου χρειάζεται εισαγωγή ενός πίνακα, έχουμε δημιουργήσει μια ειδική οθόνη (View) στην οποία ο χρήστης εισάγει τα στοιχεία στον πίνακα ένα ένα, δημιουργώντας ένα UITextField για κάθε στοιχείο του πίνακα. Αφού ο χρήστης εισάγει τις διαστάσεις του πίνακα που επιθυμεί (αριθμό γραμμών και στηλών), όταν πατήσει ένα κουμπί μιας λειτουργίας, οδηγείται στην οθόνη που φαίνεται στην εικόνα 27. Σε αυτή την οθόνη υπάρχουν τα εξής αντικείμενα:

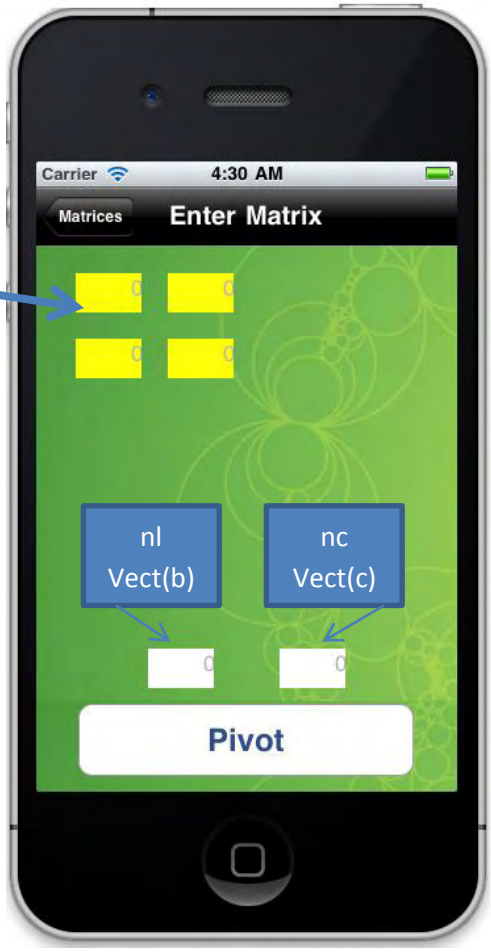
1. Μια μπάρα με τον τίτλο "Enter Matrix" στο πάνω μέρος της οθόνης και ένα κουμπί για την επιστροφή στην προηγούμενη οθόνη.
2. UIScrollView που περιέχει τα Text fields στα οποία θα εισάγει ο χρήστης τα στοιχεία του πίνακα.
3. Text field με προκαθορισμένο κείμενο "0". Όταν ο χρήστης ακουμπήσει πάνω του, εμφανίζεται πληκτρολόγιο για να εισάγει ένα στοιχείο του πίνακα. Τα Text fields δημιουργούνται δυναμικά στο UIScrollView, καθώς ο αριθμός τους δεν είναι προκαθορισμένος, αλλά δίνεται από το χρήστη στην προηγούμενη οθόνη.
4. Κουμπί για εκτέλεση της επιλεγμένης λειτουργίας. Ο τίτλος του δίνεται με βάση την επιλογή του χρήστη στην προηγούμενη οθόνη.



Εικόνα 27: Η οθόνη "Enter Matrix" για την εισαγωγή ενός πίνακα

Οι μέθοδοι “Pivot” και “Simplex” οδηγούν στην ίδια οθόνη με παραπάνω για την εισαγωγή ενός πίνακα, με τη διαφορά ότι υπάρχουν δύο επιπρόσθετα Text fields για την εισαγωγή δύο ακόμα παραμέτρων. Στην “Pivot” εισάγει ο χρήστης και ως προς ποιο οδηγό στοιχείο του πίνακα θα εκτελεστεί η παραλλαγμένη μέθοδος Gauss – Jordan, ενώ στη μέθοδο “Simplex” ζητείται από το χρήστη να εισάγει άλλα δύο διανύσματα που είναι απαραίτητα για την εκτέλεση της μεθόδου (εικόνα 28).

Εικόνα 28: Η οθόνη “Enter Matrix” για τις λειτουργίες Pivot και Simplex



Matrix A

Pivot: στοιχείο A[nl,nc] ως οδηγός

Simplex: simplex_reduce(Mtrx(A), Vect(b), Vect(c))

Τέλος, ο χρήστης μπορεί να λύσει ένα σύστημα γραμμικών εξισώσεων και να υπολογίσει τη βάση ενός χώρου στην οθόνη που οδηγείται αν πατήσει το κουμπί με τίτλο “Linear Equations”. Σε αυτή την οθόνη υπάρχουν τα εξής αντικείμενα (συγκεντρωτικά εμφανίζονται στην εικόνα 29):

1. Μια μπάρα με τον τίτλο “Linear Equations” στο πάνω μέρος της οθόνης και ένα κουμπί για την επιστροφή στην προηγούμενη (αρχική) οθόνη.
2. Text field με προκαθορισμένο κείμενο “Number of Equations or Vectors”. Όταν ο χρήστης ακουμπήσει πάνω του, εμφανίζεται πληκτρολόγιο για να εισάγει έναν ακέραιο.
3. Κουμπί για τον υπολογισμό της βάσης ενός χώρου, δοσμένου σε μορφή πίνακα, συνοδευόμενο από ένα info button για την εμφάνιση σχετικών πληροφοριών.

4. Κουμπί για την επίλυση ενός συστήματος γραμμικών εξισώσεων, συνοδευμένο από ένα info button για την εμφάνιση σχετικών πληροφοριών.

Εικόνα 29: Η οθόνη "Linear Equations"

Πατώντας το κουμπί με τίτλο "Basis", ο χρήστης οδηγείται στην οθόνη της εικόνας 30, όπου καλείται να εισάγει έναν αριθμό διανυσμάτων – ο αριθμός αυτός δόθηκε από το χρήστη στην προηγούμενη οθόνη – για να υπολογιστεί ο χώρος που παράγεται από αυτά. Τα αντικείμενα της οθόνης αυτής είναι τα ακόλουθα:

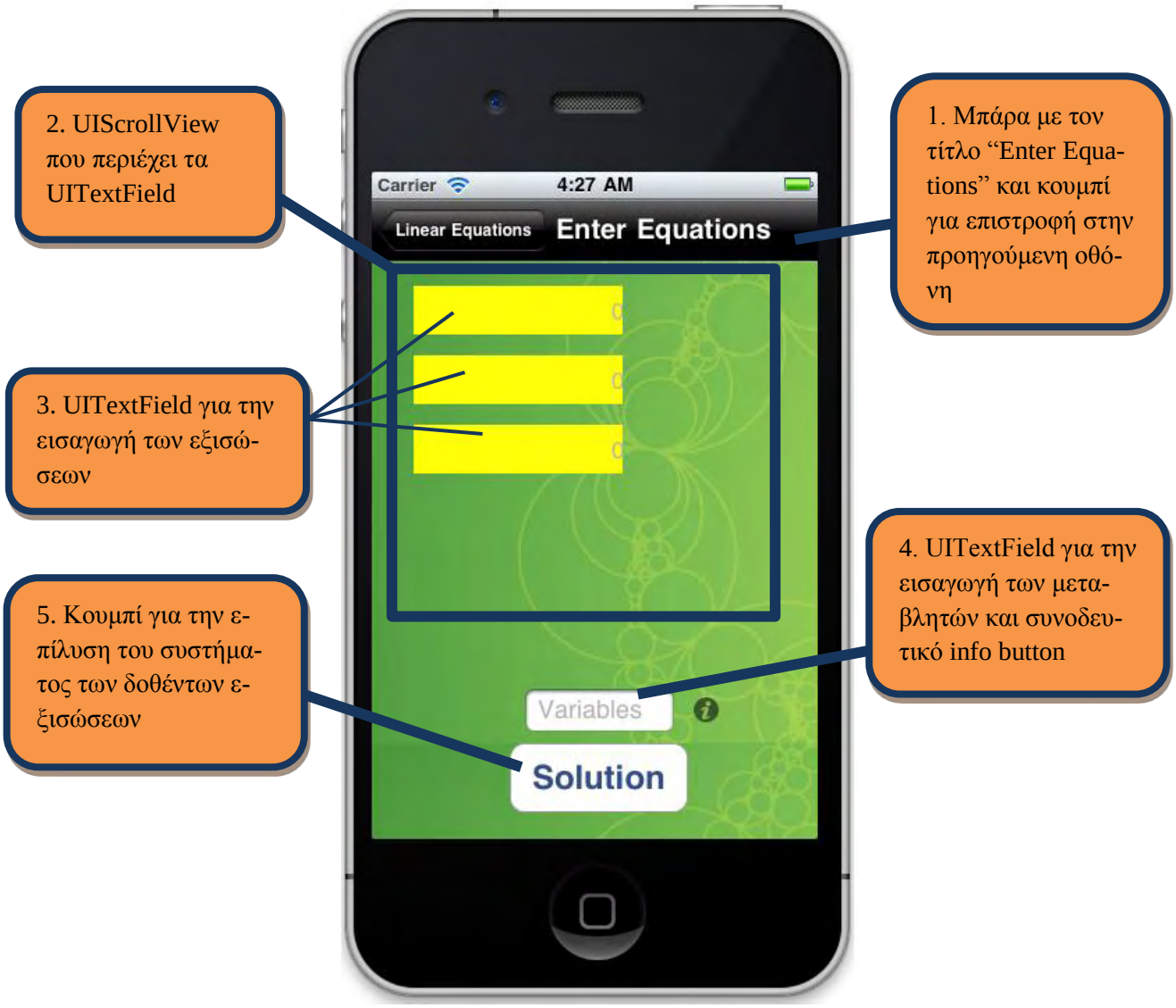
1. Μια μπάρα με τον τίτλο "Enter Vectors" στο πάνω μέρος της οθόνης και ένα κουμπί για την επιστροφή στην προηγούμενη οθόνη.
2. UIScrollView που περιέχει τα Text fields στα οποία θα εισάγει ο χρήστης τα διανύσματα, συνοδευόμενο από info button για την εμφάνιση σχετικών πληροφοριών.
3. Text fields με προκαθορισμένο κείμενο "0". Όταν ο χρήστης ακουμπήσει πάνω του, εμφανίζεται πληκτρολόγιο για να εισάγει ένα διάνυσμα. Τα Text fields δημιουργούνται δυναμικά στο UIScrollView, καθώς ο αριθμός τους δεν είναι προκαθορισμένος, αλλά δίνεται από το χρήστη στην προηγούμενη οθόνη.
4. Κουμπί για τον υπολογισμό μίας βάσης του χώρου που παράγεται από το σύνολο των δοθέντων διανυσμάτων.

Εικόνα 30: Η οθόνη "Enter Vectors"

Αν στην προηγούμενη οθόνη ο χρήστης επιλέξει τη δεύτερη επιλογή (Linear Equations), οδηγείται σε οθόνη για την εισαγωγή εξισώσεων. Αναλυτικότερα, η οθόνη περιλαμβάνει τα παρακάτω αντικείμενα (συγκεντρωτικά εμφανίζονται στην εικόνα 31):

1. Μια μπάρα με τον τίτλο "Enter Equations" στο πάνω μέρος της οθόνης και ένα κουμπί για την επιστροφή στην προηγούμενη οθόνη.
2. UIScrollView που περιέχει τα Text fields στα οποία θα εισάγει ο χρήστης τις εξισώσεις.
3. Text fields με προκαθορισμένο κείμενο "0". Όταν ο χρήστης ακουμπήσει πάνω του, εμφανίζεται πληκτρολόγιο για να εισάγει ένα διάνυσμα. Τα Text fields δημιουργούνται δυναμικά στο UIScrollView, καθώς ο αριθμός τους δεν είναι προκαθορισμένος, αλλά δίνεται από το χρήστη στην προηγούμενη οθόνη.
4. Text fields με προκαθορισμένο κείμενο "Variables" για την εισαγωγή των μεταβλητών με βάση τις οποίες θα γίνει η επίλυση του συστήματος των εξισώσεων, συνοδευόμενο από info button για τν εμφάνιση σχετικών πληροφοριών. Όταν ο χρήστης ακουμπήσει πάνω του, εμφανίζεται πληκτρολόγιο για να εισάγει τις μεταβλητές.

5. Κουμπί με τίτλο “Solution” για την επίλυση του συστήματος των δοθέντων εξισώσεων.

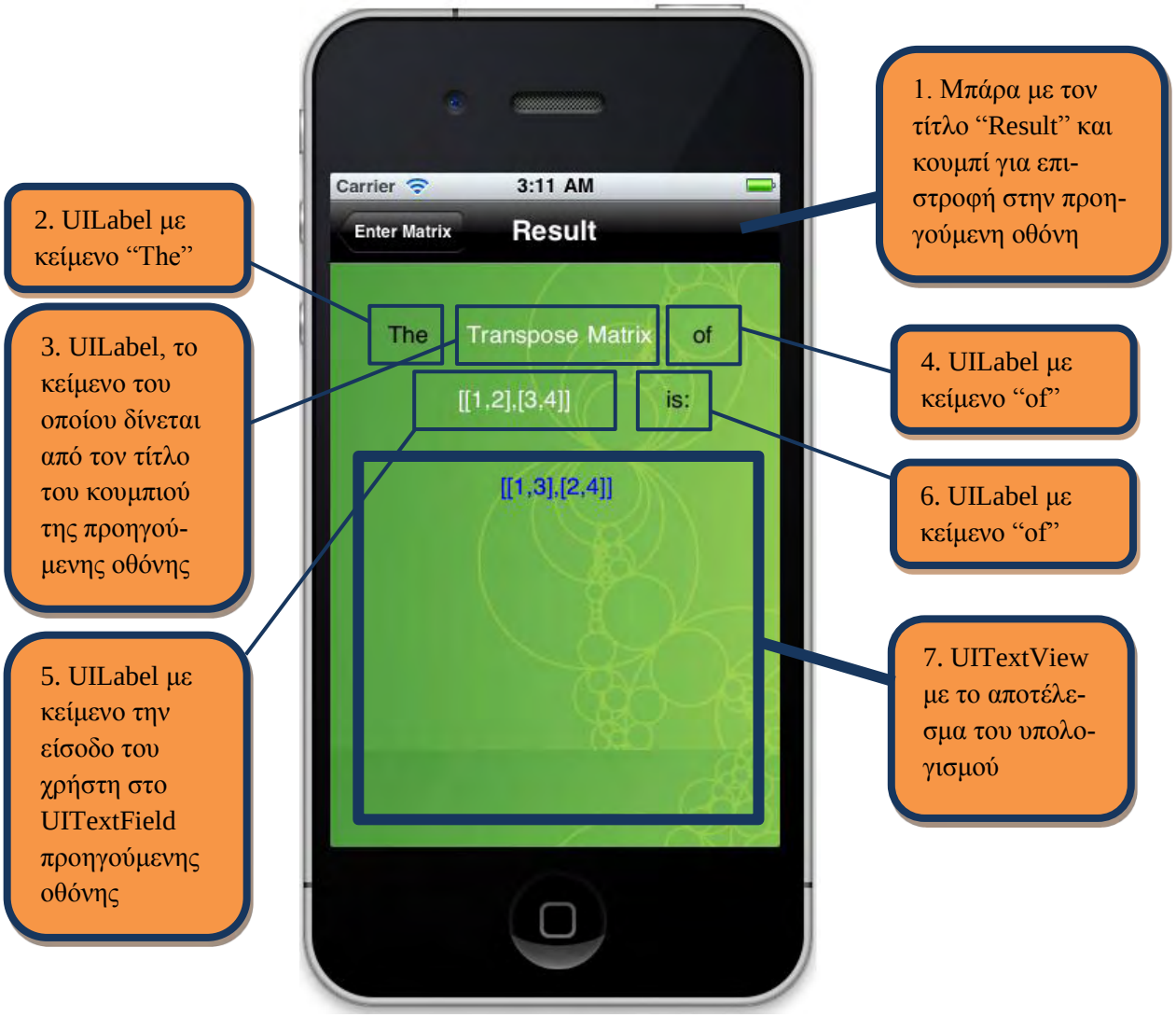


Εικόνα 31: Η οθόνη “Enter Equations”

Η οθόνη που εμφανίζονται τα αποτελέσματα από οποιαδήποτε λειτουργία είναι η ίδια για όλες τις κατηγορίες και συνίσταται από τα παρακάτω αντικείμενα (συγκεντρωτικά εμφανίζονται στην εικόνα 32):

1. Μια μπάρα με τον τίτλο “Result” στο πάνω μέρος της οθόνης και ένα κουμπί για την επιστροφή στην προηγούμενη οθόνη.
2. Ένα UILAbel με το κείμενο “The”.
3. Ένα UILabel, το κείμενο του οποίου δίνεται από τον τίτλο του κουμπιού της προηγούμενης οθόνης.
4. Ένα UILabel με το κείμενο “of”.

5. Ένα UILabel, το κείμενο του οποίου δίνεται από το τι εισάγει ο χρήστης στα UITextfield της οθόνης που προηγείται.
6. Ένα UILabel με το κείμενο “is:”.
7. Ένα UITextView που περιέχει το αποτέλεσμα του υπολογισμού.



Εικόνα 32: Η οθόνη “Resut” με το αποτέλεσμα του υπολογισμού

Εσωτερική δομή της εφαρμογής

Στην ενότητα 3.3.1 αναφέραμε το σχεδιαστικό μοντέλο MVC (Model – View – Controller) που χρησιμοποιείται από το Cocoa στις 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 μία 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 αναφέρουμε και εμφανίζει το αντίστοιχο View.

Κεφάλαιο 5ο

Συμπεράσματα

Συμπεράσματα

Μέσα από αυτή τη διπλωματική εργασία γνώρισα τις συσκευές iOS, το λειτουργικό τους σύστημα και τα προγραμματιστικά τους εργαλεία. Επίσης, αναπτύσσοντας μια εφαρμογή για μια πλατφόρμα την οποία χρησιμοποιεί σε καθημερινή βάση ένα μεγάλο μέρος του πληθυσμού, την iOS, μπόρεσα να διαπιστώσω καλύτερα τις ιδιαιτερότητες προγραμματισμού σε κινητές πλατφόρμες σε σχέση με τον προγραμματισμό σε Desktop περιβάλλοντα.

Κατά την διάρκεια της υλοποίησης της εφαρμογής βρέθηκα αρκετές φορές αντιμέτωπος με προβλήματα , όμως η διαδικασία εύρεσης λύσεων σε αυτά είναι πολύ χρήσιμη στον προγραμματισμό.

Μετά από αρκετούς μήνες ενασχόλησής μου με τον προγραμματισμό στο Apple iOS νιώθω ότι είμαι αρκετά εξοικειωμένος με την γλώσσα προγραμματισμού Objective – C αλλά και με τα εργαλεία ανάπτυξης εφαρμογών της Apple˙ και συγκεκριμένα το Xcode.

Είμαι πολύ χαρούμενος που μέσα από αυτή την πτυχιακή εργασία μού δόθηκε η ευκαιρία να ασχοληθώ με τον προγραμματισμό σε κινητές πλατφόρμες και την ανάπτυξη εφαρμογών για αυτές, καθώς είναι ένας τομέας που με ενδιαφέρει και είναι πολύ πιθανόν να ασχοληθώ επαγγελματικά με αυτόν στο μέλλον.

Μελλοντικές Επεκτάσεις

Η εφαρμογής μας, όπως και κάθε άλλο υπαρκτό πρόγραμμα, επιδέχεται μια σειρά βελτιώσεων οι οποίες θα μπορούσαν να γίνουν σε μια μελλοντική επέκτασή της. Μερικές από αυτές είναι:

- Η εισαγωγή δύο πινάκων με δυναμικά UITextField και σώσιμο πίνακα

Όπως μπορεί να δει κανείς στην εικόνα 25, για να εισάγει ο χρήστης δύο πίνακες, πρέπει να τους εισάγει με τη μορφή

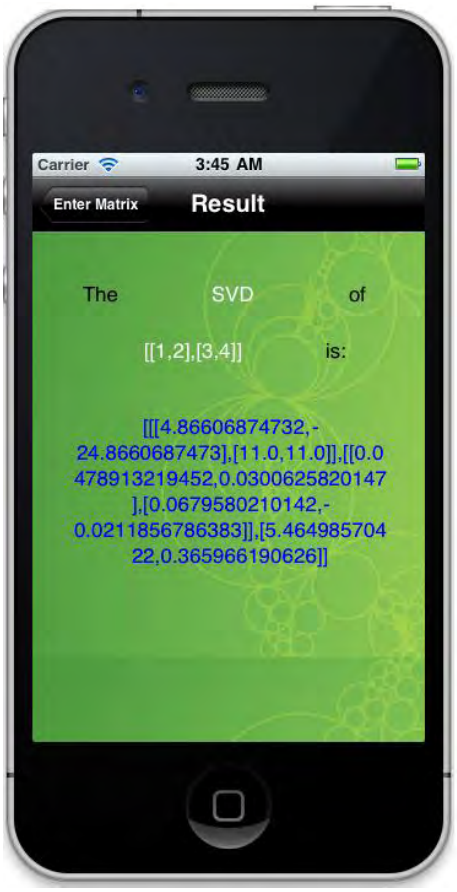
[[στοιχείο1,στοιχείο2,…,στοιχείοN],…,[στοιχείο1,στοιχείο2,…,στοιχείοN]]

και όχι εισάγοντας ένα ένα τα στοιχεία (εικόνα 27). Έτσι, μια βελτίωση θα ήταν ο χρήστης να εισάγει τον αριθμό γραμμών και στηλών των δύο πινάκων, να εισάγει τον πρώτο πίνακα στοιχείο στοιχείο, να σώζει τον πίνακα, να εισάγει και να σώζει το δεύτερο πίνακα με τον ίδιο

τρόπο και τέλος, να εκτελείται η πράξη μεταξύ τους. Με αυτό τον τρόπο δε χρειάζεται να γίνεται έλεγχος ότι οι πίνακες που έχει εισάγει ο χρήστης είναι ίδιας διάστασης.

- Βελτίωση της εμφάνισης του αποτελέσματος

Σε κάποιες περιπτώσεις όπου το αποτέλεσμα που προκύπτει είναι μεγάλο σε μέγεθος και χρειάζονται περισσότερες από μία γραμμές για να εμφανιστεί, η εμφάνιση δεν είναι η καλύτερη δυνατή (εικόνα 5.1) και πρέπει να γίνουν κάποιες τροποποιήσεις στην εφαρμογή ώστε να αλλάξει αυτό.



Εικόνα 33: Άσχημη εμφάνιση αποτελέσματος

- Να σώζεται ο πίνακας που έχει εισάγει ο χρήστης σε περίπτωση αναγκαστικού κλεισίματος της εφαρμογής (crash)

Υπάρχουν περιπτώσεις όπου μια εφαρμογή που τρέχει σε μια φορητή συσκευή να «κολλήσει» και να χρειάζεται αναγκαστικό κλείσιμο. Αυτό μπορεί να συμβεί είτε γιατί η εφαρμογή έχει κάποιο πρόβλημα είτε γιατί ευθύνεται η ίδια η συσκευή. Σε κάθε περίπτωση, θα ήταν ευκολία για το χρήστη να μη χρειάζεται να εισάγει τον πίνακα από την αρχή, αλλά να είναι διαθέσιμος για χρήση όταν ανοίξει ξανά την εφαρμογή.

- Περισσότερες προφυλάξεις από κακόβουλο (ή απρόσεχτο) χρήστη

Στην εφαρμογή μας έχουμε μεριμνήσει για εισαγωγή λανθασμένων δεδομένων από το χρήστη (κενό text field, αρνητικός αριθμός γραμμών/στηλών πίνακα, εισαγωγή γράμματος αντί για αριθμό, κ.ά.), αλλά υπάρχουν επιπρόσθετες περιπτώσεις που χρήζουν προσοχής και τις οποίες πρόκειται να λάβουμε υπόψη στο μέλλον.

Κεφάλαιο 6ο

Βιβλιογραφία

Εικόνες

[1] Basic_1. 18 Ιουνίου 2012 HiCalc PRO An ultimate calculator for iPhone Basic Screenshots, PPCLINK Apps: http://www.ppclink.com/apps/index.php?page=hicalc-pro-basic

[2] green-back2.jpg. 12 Φεβρουαρίου 2012 Background image, Wolfram Alpha: http://www.wolframcdn.com/homepage/green-back2.jpg

[3] iPhone_Alpha_LinearAlgebra1. 17 Απριλίου 2012 Linear Algebra Course Assistant Screenshots for iPhone, Wolfram Alpha Mobile & Tablet Apps: http://products.wolframalpha.com/courseassistants/linear-algebra.html

[4] iPhone_Alpha_LinearAlgebra3. 17 Απριλίου 2012 Linear Algebra Course Assistant Screenshots for iPhone, Wolfram Alpha Mobile & Tablet Apps: http://products.wolframalpha.com/courseassistants/linear-algebra.html

[5] mza_15049179663688251.320x480-75. 18 Ιουνίου 2012 PocketCAS 3 pro Mathematics Suite Screenshots for iPhone, iTunes: http://itunes.apple.com/app/id517250782?mt=8

[6] mza_7322178714685209448.320x480-75. 18 Ιουνίου 2012 PocketCAS 3 pro Mathematics Suite Screenshots for iPhone, iTunes: http://itunes.apple.com/app/id517250782?mt=8

[7] scientific01. 18 Ιουνίου 2012 Calc Pro Screenshots for iPhone, Panoramic Software Inc.: http://www.panoramicsoft.com/iphone/calcpro/CalcPro.php

[8] unnamed. 18 Ιουνίου 2012 Linear Algebra Study Guide Screenshots, Google Play: https://play.google.com/store/apps/details?id=com.mobilereference.linearalgebra&featu-re=search_result#?t=W251bGwsMSwxLDEsImNvbS5tb2JpbGVyZWZlcmVuY2UubGluZWFyYWxnZWJyYSJd

[9] unnamed. 18 Ιουνίου 2012 Matrix Calculator Screenshots, Google Play: https://play.google.com/store/apps/details?id=com.bogeyman.matrixfree&feature=search_result#?t=W251bGwsMSwxLDEsImNvbS5ib2dleW1hbi5tYXRyaXhmcmVlIl0

[10] unnamed. 18 Ιουνίου 2012 Pocket Linear Algebra Tutor Screenshots, Google Play: https://play.google.com/store/apps/details?id=org.plat.plat&feature=search_result#?t=W251bGwsMSwxLDEsIm9yZy5wbGF0LnBsYXQiXQ..

Κείμενο και γραμμές κώδικα

[1] Apple Developer Tools presentation: https://developer.apple.com/technologies/tools/

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

[3] Stack Overflow Developer Community: http://stackoverflow.com/

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

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

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

[7] Wikipedia. Objective – C article: http://en.wikipedia.org/wiki/Objective-C

Παράρτημα

PsychologistViewController.h

```
#import <UIKit/UIKit.h>


@interface PsychologistViewController : UIViewController {

    IBOutlet UIView *portrait;
    IBOutlet UIView *landscape;

}

-(IBAction)vectors;
-(IBAction)matrix;
-(IBAction)linear;
-(IBAction)factorization;

@property(nonatomic,retain) IBOutlet UIView *portrait;
@property(nonatomic,retain) IBOutlet UIView *landscape;

@end
```

PsychologistViewController.m

```
#import "PsychologistViewController.h"
#import "FactorizationViewController.h"
#import "TestTwoFields.h"
#import "ThreeFields.h"
#import "FactorViewController.h"


@implementation PsychologistViewController

@synthesize portrait;
@synthesize landscape;

-(void)showVectors
{
    TestTwoFields *tapvc = [[TestTwoFields alloc] init];
    tapvc.title = @"Vectors";
    [self.navigationController pushViewController:tapvc animat-
ed:YES];
    [tapvc release];
}

-(void)showMatrix
{
    FactorViewController *tapvc = [[FactorViewController alloc] in-
it];
    tapvc.title = @"Matrices";
    [self.navigationController pushViewController:tapvc animat-
ed:YES];
```

```
    [tapvc release];
}

-(void)showLinear
{
    ThreeFields *tapvc = [[ThreeFields alloc] init];
    tapvc.title = @"Linear Equations";
    [self.navigationController pushViewController:tapvc animat-
ed:YES];
    [tapvc release];
}

-(void)showFactor
{
    FactorizationViewController *tapvc = [[FactorizationViewControl-
ler alloc] init];
    tapvc.title = @"Factorization";
    [self.navigationController pushViewController:tapvc animat-
ed:YES];
    [tapvc release];
}


-(IBAction)vectors
{
    [self showVectors];
}

-(IBAction)matrix
{
    [self showMatrix];
}

-(IBAction)linear
{
    [self showLinear];
}

-(IBAction)factorization
{
    [self showFactor];
}


- (void) showGeneralInfoView:(id)sender {
    UIAlertView *generalInfoAlert = [[UIAlertView alloc]
                                     initWithTitle:@"Linear Algebra
App"
                                     message:@"This application is a
computer algebra module for Linear Algebra operations.\n\nThe app was
created by Spyros Chasakos, 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.\n\nThank you for
using our app. Please send us your feedback to akritas@uth.gr"
                                     delegate:nil
                                     cancelButtonTitle:@"OK"
                                     otherButtonTitles:nil];
    [generalInfoAlert show];
    [generalInfoAlert release];
```

```
}

// The designated initializer.  Override if you create the controller
programmatically and want to perform customization that is not appro-
priate for viewDidLoad.
/*
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle
*)nibBundleOrNil {
    self = [super initWithNibName:nibNameOrNil bun-
dle:nibBundleOrNil];
    if (self) {
        // Custom initialization.
    }
    return self;
}
*/


// Implement viewDidLoad to do additional setup after loading the
view, typically from a nib.
- (void)viewDidLoad {
    [super viewDidLoad];
    self.view.backgroundColor = [UIColor clearColor];
    landscape.backgroundColor = [UIColor clearColor];

    //Navigation Bar info button
    UIButton* generalInfoButton = [UIButton button-
WithType:UIButtonTypeInfoLight];
    [generalInfoButton addTarget:self ac-
tion:@selector(showGeneralInfoView:) forControlEv-
ents:UIControlEventTouchUpInside];
    UIBarButtonItem *barButtonItem = [[UIBarButtonItem alloc] init-
WithCustomView:generalInfoButton];
    self.navigationItem.rightBarButtonItem = barButtonItem;
    [barButtonItem release];
}



// Override to allow orientations other than the default portrait
orientation.
-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)
interfaceOrientation {
    // Return YES for supported orientations.
    //return (interfaceOrientation == UIInterfaceOrientationLand-
scapeLeft);
    if(((interfaceOrientation == UIInterfaceOrientationLandscapeLeft)
||
        (interfaceOrientation == UIInterfaceOrientationLand-
scapeRight))) {
        self.view = landscape;
    }
    else if (((interfaceOrientation == UIInterfaceOrientationPor-
trait) ||
              (interfaceOrientation == UIInterfaceOrientationPortrai-
tUpsideDown))) {
        self.view = portrait;
    }
    return YES;
}
```

```
- (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;
}

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


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


@end
```

VectorTwoFields.h

```
#import <UIKit/UIKit.h>
#import "TestAddProBrain2.h"


@interface VectorTwoFields : UIViewController {
    TestAddProBrain2 *brain;
    IBOutlet UITextField *textField;
    IBOutlet UITextField *textField2;
    IBOutlet UIView *portrait;
    IBOutlet UIView *landscape;
    NSString *result;

}

@property(nonatomic,retain) IBOutlet UITextField *textField;
@property(nonatomic,retain) IBOutlet UITextField *textField2;
@property(nonatomic,retain) IBOutlet UIView *portrait;
@property(nonatomic,retain) IBOutlet UIView *landscape;
@property (nonatomic,retain) NSString *result;


-(IBAction)buttonPressed:(UIButton *)sender;
-(IBAction)textFieldDoneEditing:(id)sender;
-(IBAction)infoButtonPressed:(UIButton *)sender;
-(IBAction)infoButtonPressed2:(UIButton *)sender;
```

```
@end
```

VectorTwoFields.m

```
#import "VectorTwoFields.h"
#import "TestAddProBrain2.h"
#import "FurtherView.h"


@implementation VectorTwoFields

@synthesize textField;
@synthesize textField2;
@synthesize result;
@synthesize portrait;
@synthesize landscape;

-(TestAddProBrain2 *)brain
{
    if (!brain) brain = [[TestAddProBrain2 alloc] init];
    return brain;
}

-(BOOL)stringIsVector{

    NSLog(@"I'm in with %@",textField.text);
    NSString *string = textField.text;

    NSString *pattern = @"\\[|\\(";

    NSError *error = NULL;
    NSRegularExpression *regex = [NSRegularExpression regularExpres-
sionWithPattern:pattern

options:NSRegularExpressionCaseInsensitive

error:&error];

    NSArray* matches = [regex matchesInString:string options:0
range:NSMakeRange(0, [string length])];
    for ( NSTextCheckingResult* match in matches )
    {
        NSString* matchText = [string substringWithRange:[match
range]];
        NSLog(@"match: %@", matchText);
        if (string.length == matchText.length) {
            return YES;
        }
    }

    return NO;
}

-(void)showMagic
{
    FurtherView *tapvc = [[FurtherView alloc] init];
    tapvc.title = @"Result";
```

```
    [tapvc setResult:self.result];
    [self.navigationController pushViewController:tapvc animat-
ed:YES];
    [tapvc release];
}


-(IBAction)buttonPressed:(UIButton *)sender
{
    if(([textField.text length]==0)||([textField2.text length]==0)){
        UIAlertView *noInput = [[UIAlertView alloc]
                                initWithTitle:@"Invalid Input"
                                message:@"No vector inserted. Please
insert a vector."
                                delegate:self
                                cancelButtonTitle:@"OK"
                                otherButtonTitles:nil];
        [noInput show];
        [noInput release];
    }
    else {
        //if([self stringIsVector]){
            NSString *todo = [[sender titleLabel] text];
            NSString *operation = [[NSString alloc] initWithFor-
mat:@"%@",textField.text];
            NSString *tnt2 = [[NSString alloc] initWithFor-
mat:@"%@",textField2.text];
            self.result = [[self brain] lala3:operation sasa:tnt2 ma-
sa:todo];
            [self showMagic];
        /*}
        else {
            UIAlertView *invalidInput = [[UIAlertView alloc]
                                         initWithTitle:@"Invalid In-
put"
                                         message:@"What you entered
is not a vector. Please insert a correct vector."
                                         delegate:self
                                         cancelButtonTitle:@"OK"
                                         otherButtonTitles:nil];
            [invalidInput show];
            [invalidInput release];
        }*/

    }
}

-(IBAction)textFieldDoneEditing:(id)sender
{
    [sender resignFirstResponder];
}

-(IBAction)infoButtonPressed:(UIButton *)sender
{
    UIAlertView *instructionsAlert = [[UIAlertView alloc]
                                      initWithTitle:@"Cross Product"
                                      message:@"Wedge prod-
uct.\n\ncross(Vect,Vect)"
                                      delegate:nil
                                      cancelButtonTitle:@"OK"
                                      otherButtonTitles:nil];
```

```
    [instructionsAlert show];
    [instructionsAlert release];
}

-(IBAction)infoButtonPressed2:(UIButton *)sender
{
    UIAlertView *instructionsAlert = [[UIAlertView alloc]
                                      initWithTitle:@"Dot product"
                                      message:@"Scalar prod-
uct.\n\ndot(Vect,Vect)"
                                      delegate:nil
                                      cancelButtonTitle:@"OK"
                                      otherButtonTitles:nil];
    [instructionsAlert show];
    [instructionsAlert release];
}

// The designated initializer.  Override if you create the controller
programmatically and want to perform customization that is not appro-
priate for viewDidLoad.
/*
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle
*)nibBundleOrNil {
    self = [super initWithNibName:nibNameOrNil bun-
dle:nibBundleOrNil];
    if (self) {
        // Custom initialization.
    }
    return self;
}
*/


// Implement viewDidLoad to do additional setup after loading the
view, typically from a nib.
- (void)viewDidLoad {
    [super viewDidLoad];
    self.view.backgroundColor = [UIColor clearColor];
    landscape.backgroundColor = [UIColor clearColor];
    textField.keyboardType = UIKeyboardTypeNumbersAndPunctuation;
    textField2.keyboardType = UIKeyboardTypeNumbersAndPunctuation;
}



// Override to allow orientations other than the default portrait
orientation.
-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)
interfaceOrientation {
    // Return YES for supported orientations.
    if(((interfaceOrientation == UIInterfaceOrientationLandscapeLeft)
||
        (interfaceOrientation == UIInterfaceOrientationLand-
scapeRight))) {
        self.view = landscape;
        [landscape addSubview:textField];
        [landscape addSubview:textField2];
    }
    else if (((interfaceOrientation == UIInterfaceOrientationPor-
trait) ||
```

```
              (interfaceOrientation == UIInterfaceOrientationPortrai-
tUpsideDown))) {
        self.view = portrait;
        [portrait addSubview:textField];
        [portrait addSubview:textField2];
    }
    return YES;
}


- (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.textField = nil;
    self.textField2 = nil;
}

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


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


@end
```

LinearEquationsViewController.h

```
#import <UIKit/UIKit.h>
#import "TestAddProBrain2.h"


@interface LinearEquationsViewController : UIViewController {

    TestAddProBrain2 *brain;
    int noOfEquations;
    IBOutlet UITextField *field2;
    NSString *myunion;
    NSString *result;
    NSMutableArray *textFields;
    IBOutlet UIScrollView *display;
    IBOutlet UIView *portrait;
    IBOutlet UIView *landscape;
```

```
}

@property (nonatomic,assign) int noOfEquations;
@property(nonatomic,retain) IBOutlet UITextField *field2;
@property (retain) IBOutlet UIScrollView *display;
@property (nonatomic,retain) NSString *result;
@property(nonatomic,retain) IBOutlet UIView *portrait;
@property(nonatomic,retain) IBOutlet UIView *landscape;

-(IBAction)buttonPressed:(UIButton *)sender;
-(IBAction)textFieldDoneEditing:(id)sender;
-(NSString *)checkEmptyTextFields;
-(IBAction)infoButtonPressed:(UIButton *)sender;

@end
```

LinearEquationsViewController.m

```
#import "LinearEquationsViewController.h"
#import "FurtherView.h"


@implementation LinearEquationsViewController

@synthesize noOfEquations;
@synthesize result;
@synthesize field2;
@synthesize display;
@synthesize portrait;
@synthesize landscape;

-(TestAddProBrain2 *)brain
{
    if (!brain) brain = [[TestAddProBrain2 alloc] init];
    return brain;
}

-(void)showMagic
{
    FurtherView *tapvc = [[FurtherView alloc] init];
    tapvc.title = @"Result";
    [tapvc setResult:self.result];
    [self.navigationController pushViewController:tapvc animat-
ed:YES];
    [tapvc release];
}


-(IBAction)buttonPressed:(UIButton *)sender
{
    NSString *matrix = [self checkEmptyTextFields];


    if([matrix isEqual:@"Empty"])
    {
        UIAlertView *noInput = [[UIAlertView alloc]
                                initWithTitle:@"Invalid Input"
```

```
                                message:@"At least one of the fields
is empty"
                                delegate:self
                                cancelButtonTitle:@"OK"
                                otherButtonTitles:nil];
        [noInput show];
        [noInput release];
    }
    else if([matrix isEqual:@"Invalid"])
    {
        UIAlertView *invalid = [[UIAlertView alloc]
                                initWithTitle:@"Invalid Input"
                                message:@"At least one of the equa-
tions you entered is wrong"
                                delegate:self
                                cancelButtonTitle:@"OK"
                                otherButtonTitles:nil];
        [invalid show];
        [invalid release];
    }
    else {
        NSLog(@"Matrix %@",matrix);
        NSLog(@"%@",field2.text);
        matrix = [matrix stringByAppendingString:@",["];
        matrix = [matrix stringByAppendingString:field2.text];
        matrix = [matrix stringByAppendingString:@"]"];
        self.result = [[self brain] solveLinear:matrix];
        [self showMagic];
    }

}

-(IBAction)textFieldDoneEditing:(id)sender
{
    [sender resignFirstResponder];
}

-(NSString *)checkEmptyTextFields
{
    myunion = [[NSString alloc] initWithFormat:@"["];

    for(UITextField *spyros in textFields)
    {
        if ([spyros.text length]== 0) {
            return @"Empty";
        }
        else {
            myunion = [myunion stringByAppendingString:spyros.text];
            myunion = [myunion stringByAppendingString:@","];
        }
    }
    myunion = [myunion substringToIndex:[myunion length] - 1];
    myunion = [myunion stringByAppendingString:@"]"];
    NSLog(@"%@",myunion);
    return myunion;
}


-(void)createTextField
{
```

```
    textFields = [[NSMutableArray alloc] init];
    [display setContentSize:CGSizeMake(150,noOfEquations*50)];

    for(int i=10;i<noOfEquations*50+10;i+=50)
    {
        UITextField *field = [[UITextField alloc] initWith-
Frame:CGRectMake(10, i, 150, 35)];
        field.returnKeyType = UIReturnKeyDone;
        field.placeholder = @"0";
        field.autocapitalizationType = UITextAutocapitalizationType-
Words;
        field.adjustsFontSizeToFitWidth = TRUE;
        field.backgroundColor = [UIColor yellowColor];
        field.keyboardType = UIKeyboardTypeNumbersAndPunctuation;
        field.textAlignment = UITextAlignmentRight;
        field.contentVerticalAlignment = UIControlContentVerticalA-
lignmentCenter;
        [field addTarget:self
                  action:@selector(textFieldDone:)
        forControlEvents:UIControlEventEditingDidEndOnExit];
        [textFields addObject:field];
        [display addSubview:field];
        [field release];
    }
    display.scrollEnabled = YES;

}

-(IBAction)infoButtonPressed:(UIButton *)sender
{
    UIAlertView *instructionsAlert = [[UIAlertView alloc]
                                      initWithTitle:@"Variables"
                                      message:@"Insert the variables
of the equations separated by comma"
                                      delegate:nil
                                      cancelButtonTitle:@"OK"
                                      otherButtonTitles:nil];
    [instructionsAlert show];
    [instructionsAlert release];
}

- (IBAction)textFieldDone:(id)sender {
    [sender resignFirstResponder];
}


// The designated initializer.  Override if you create the controller
programmatically and want to perform customization that is not appro-
priate for viewDidLoad.
/*
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle
*)nibBundleOrNil {
    self = [super initWithNibName:nibNameOrNil bun-
dle:nibBundleOrNil];
    if (self) {
        // Custom initialization.
    }
    return self;
}
*/
```

```
// Implement viewDidLoad to do additional setup after loading the
view, typically from a nib.
- (void)viewDidLoad {
    [super viewDidLoad];
    self.view.backgroundColor = [UIColor clearColor];
    display.backgroundColor = [UIColor clearColor];
    [self createTextField];
}


/*
// Override to allow orientations other than the default portrait
orientation.
-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)
interfaceOrientation {
    // Return YES for supported orientations.
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}
*/

- (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.field2 = nil;
    self.landscape = nil;
    self.portrait = nil;
    self.display = nil;
}

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


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


@end
```

PivotViewController.h

```
#import <UIKit/UIKit.h>
#import "TestAddProBrain2.h"
```

```
@interface PivotViewController : UIViewController {

    TestAddProBrain2 *brain;
    IBOutlet UITextField *field2;
    IBOutlet UITextField *field3;
    NSString *result;
    NSString *todo;
    NSString *myunion;
    NSMutableArray *textFields;
    IBOutlet UIScrollView *display;
    IBOutlet UIScrollView *display2;
    IBOutlet UIScrollView *display3;
    IBOutlet UIScrollView *display4;
    IBOutlet UIButton *operation;
    IBOutlet UIButton *operation2;
    int rows;
    int columns;
    int noOfEquations;
    IBOutlet UIView *portrait;
    IBOutlet UIView *landscape;
    int times;
    NSString *takeOperation;
    NSString *takeInput;
}

@property(nonatomic,retain) IBOutlet UITextField *field2;
@property(nonatomic,retain) IBOutlet UITextField *field3;
@property(nonatomic,assign) int rows;
@property(nonatomic,assign) int columns;
@property (nonatomic,retain) NSString *result;
@property (nonatomic,retain) NSString *todo;
@property (retain) IBOutlet UIScrollView *display;
@property (retain) IBOutlet UIScrollView *display2;
@property (retain) IBOutlet UIScrollView *display3;
@property (retain) IBOutlet UIScrollView *display4;
@property(nonatomic,assign) IBOutlet UIButton *operation;
@property(nonatomic,assign) IBOutlet UIButton *operation2;
@property (nonatomic,assign) int noOfEquations;
@property(nonatomic,retain) IBOutlet UIView *portrait;
@property(nonatomic,retain) IBOutlet UIView *landscape;
@property (nonatomic,retain) NSString *takeOperation;
@property (nonatomic,retain) NSString *takeInput;

-(IBAction)buttonPressed:(UIButton *)sender;
-(IBAction)textFieldDoneEditing:(id)sender;
-(NSString *)checkEmptyTextFields;

@end
```

PivotViewController.m

```
#import "PivotViewController.h"
#import "TestAddProBrain2.h"
#import "FurtherView.h"
```

```
@implementation PivotViewController

@synthesize field2;
@synthesize field3;
@synthesize result;
@synthesize display;
@synthesize display2;
@synthesize display3;
@synthesize display4;
@synthesize rows;
@synthesize columns;
@synthesize operation;
@synthesize todo;
@synthesize noOfEquations;
@synthesize portrait;
@synthesize landscape;
@synthesize operation2;
@synthesize takeOperation;
@synthesize takeInput;


-(TestAddProBrain2 *)brain
{
    if (!brain) brain = [[TestAddProBrain2 alloc] init];
    return brain;
}

-(void)showMagic
{
    FurtherView *tapvc = [[FurtherView alloc] init];
    tapvc.title = @"Result";
    [tapvc setResult:self.result];
    [tapvc setTakeInput:self.takeInput];
    [tapvc setTakeOperation:self.takeOperation];
    [self.navigationController pushViewController:tapvc animat-
ed:YES];
    [tapvc release];
}


-(IBAction)buttonPressed:(UIButton *)sender
{
    NSString *matrix = [self checkEmptyTextFields];
    if([matrix isEqual:@"Empty"])
    {
        UIAlertView *noInput = [[UIAlertView alloc]
                                initWithTitle:@"Invalid Input"
                                message:@"One of the fields is empty"
                                delegate:self
                                cancelButtonTitle:@"OK"
                                otherButtonTitles:nil];
        [noInput show];
        [noInput release];
    }
    else {
        if (([todo isEqual:@"Pivot"])||([todo isEqual:@"Simplex Meth-
od"])) {
            self.result = [[self brain] matrix:myunion
bvec:field2.text cvec:field3.text operation:todo];
        }
        else
```

```
        {
            self.result = [[self brain] lala2:myunion what:[[sender
titleLabel] text]];
        }
        self.takeOperation = todo;
        self.takeInput = myunion;
        [self showMagic];
    }

}

-(IBAction)textFieldDoneEditing:(id)sender
{
    [sender resignFirstResponder];
}

-(NSString *)checkEmptyTextFields
{
    myunion = [[NSString alloc] initWithFormat:@"["];
    int jj = 1;

    for(UITextField *spyros in textFields)
    {
        if (jj==1) {
            myunion = [myunion stringByAppendingString:@"["];
        }
        NSLog(@"%@",spyros.text);
        if ([spyros.text length]== 0) {
            return @"Empty";
        }
        else {

            myunion = [myunion stringByAppendingString:spyros.text];
            myunion = [myunion stringByAppendingString:@","];

        }
        if (jj==columns) {
            myunion = [myunion substringToIndex:[myunion length] -
1];
            jj=0;
            myunion = [myunion stringByAppendingString:@"],"];
        }
        jj++;

    }
    myunion = [myunion substringToIndex:[myunion length] - 1];
    myunion = [myunion stringByAppendingString:@"]"];
    NSLog(@"%@",myunion);
    return myunion;

}

-(void)createTextField
{
    times++;

    NSLog(@"Create Textfield with times");

    textFields = [[NSMutableArray alloc] init];

    [display setContentSize:CGSizeMake(columns*70,rows*50)];
```

```
    [display3 setContentSize:CGSizeMake(columns*70,rows*50)];

    for(int i=10;i<rows*50+10;i+=50)
    {
        for(int jj=10;jj<columns*70+10;jj+=70)
        {

            UITextField *field = [[UITextField alloc] initWith-
Frame:CGRectMake(jj, i, 50, 30)];
            field.returnKeyType = UIReturnKeyDone;
            field.placeholder = @"0";
            field.autocapitalizationType = UITextAutocapitalization-
TypeWords;
            field.adjustsFontSizeToFitWidth = TRUE;
            field.backgroundColor = [UIColor yellowColor];
            field.keyboardType = UIKeyboardTypeNumbersAndPunctuation;
            field.textAlignment = UITextAlignmentRight;
            [field addTarget:self
                          action:@selector(textFieldDone:)
                forControlEvents:UIControlEventEditingDidEndOnExit];
            [textFields addObject:field];

            if(((self.interfaceOrientation == UIInterfaceOrientation-
Portrait) ||
                (self.interfaceOrientation == UIInterfaceOrientation-
PortraitUpsideDown))) {

                [display addSubview:field];
                if (times==2) {
                    [display3 addSubview:field];
                }
            }
            else if(((self.interfaceOrientation == UIInterfaceOrien-
tationLandscapeLeft) ||
                     (self.interfaceOrientation == UIInterfaceOrien-
tationLandscapeRight))) {
                [display3 addSubview:field];
            }

            [field release];
        }
    }
    NSLog(@"%@ ---------",todo);
    if (([todo isEqual:@"Pivot"])||([todo isEqual:@"Simplex Meth-
od"])) { //Two extra fields for pivot and simplex
        NSLog(@"EImai maesa");

        [display2 setContentSize:CGSizeMake(200,100)];
        [display4 setContentSize:CGSizeMake(150,50)];

        if(((self.interfaceOrientation == UIInterfaceOrientationPor-
trait) ||
            (self.interfaceOrientation == UIInterfaceOrientationPor-
traitUpsideDown))) {
            field2 = [[UITextField alloc] initWith-
Frame:CGRectMake(45, 75, 50, 30)];
            if (times==2) {
                field2 = [[UITextField alloc] initWith-
Frame:CGRectMake(10, 30, 50, 30)];
            }
        }
```

```
        else if(((self.interfaceOrientation == UIInterfaceOrienta-
tionLandscapeLeft) ||
                 (self.interfaceOrientation == UIInterfaceOrienta-
tionLandscapeRight))) {
            field2 = [[UITextField alloc] initWith-
Frame:CGRectMake(10, 30, 50, 30)];
        }


        field2.returnKeyType = UIReturnKeyDone;
        field2.placeholder = @"0";
        field2.autocapitalizationType = UITextAutocapitalizationType-
Words;
        field2.adjustsFontSizeToFitWidth = TRUE;
        field2.backgroundColor = [UIColor whiteColor];
        field2.keyboardType = UIKeyboardTypeNumbersAndPunctuation;
        field2.textAlignment = UITextAlignmentRight;
        [field2 addTarget:self
                   action:@selector(textFieldDone:)
        forControlEvents:UIControlEventEditingDidEndOnExit];
        if(((self.interfaceOrientation == UIInterfaceOrientationPor-
trait) ||
            (self.interfaceOrientation == UIInterfaceOrientationPor-
traitUpsideDown))) {
            [display2 addSubview:field2];
            if (times==2) {
                [display4 addSubview:field2];
            }
        }
        else if(((self.interfaceOrientation == UIInterfaceOrienta-
tionLandscapeLeft) ||
                 (self.interfaceOrientation == UIInterfaceOrienta-
tionLandscapeRight))) {
            [display4 addSubview:field2];
        }
        [field2 release];


        if(((self.interfaceOrientation == UIInterfaceOrientationPor-
trait) ||
            (self.interfaceOrientation == UIInterfaceOrientationPor-
traitUpsideDown))) {
            field3 = [[UITextField alloc] initWith-
Frame:CGRectMake(145, 75, 50, 30)];
            if (times==2) {
                field3 = [[UITextField alloc] initWith-
Frame:CGRectMake(100, 30, 50, 30)];
            }
        }
        else if(((self.interfaceOrientation == UIInterfaceOrienta-
tionLandscapeLeft) ||
                 (self.interfaceOrientation == UIInterfaceOrienta-
tionLandscapeRight))) {
            field3 = [[UITextField alloc] initWith-
Frame:CGRectMake(100, 30, 50, 30)];
        }



        field3.returnKeyType = UIReturnKeyDone;
        field3.placeholder = @"0";
```

```
        field3.autocapitalizationType = UITextAutocapitalizationType-
Words;
        field3.adjustsFontSizeToFitWidth = TRUE;
        field3.backgroundColor = [UIColor whiteColor];
        field3.keyboardType = UIKeyboardTypeNumbersAndPunctuation;
        field3.textAlignment = UITextAlignmentRight;
        [field3 addTarget:self
                   action:@selector(textFieldDone:)
        forControlEvents:UIControlEventEditingDidEndOnExit];
        if(((self.interfaceOrientation == UIInterfaceOrientationPor-
trait) ||
            (self.interfaceOrientation == UIInterfaceOrientationPor-
traitUpsideDown))) {
            [display2 addSubview:field3];
            if (times==2) {
                [display4 addSubview:field3];
            }
        }
        else if(((self.interfaceOrientation == UIInterfaceOrienta-
tionLandscapeLeft) ||
                 (self.interfaceOrientation == UIInterfaceOrienta-
tionLandscapeRight))) {
            [display4 addSubview:field3];
        }

        [field3 release];
    }
        display.scrollEnabled =YES;
        display2.scrollEnabled =YES;
        display3.scrollEnabled =YES;
        display4.scrollEnabled =YES;
}


- (IBAction)textFieldDone:(id)sender {
    [sender resignFirstResponder];
}




// The designated initializer.  Override if you create the controller
programmatically and want to perform customization that is not appro-
priate for viewDidLoad.
/*
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle
*)nibBundleOrNil {
    self = [super initWithNibName:nibNameOrNil bun-
dle:nibBundleOrNil];
    if (self) {
        // Custom initialization.
    }
    return self;
}
*/


// Implement viewDidLoad to do additional setup after loading the
view, typically from a nib.
- (void)viewDidLoad {
    [super viewDidLoad];
```

```
    times=0;
    self.view.backgroundColor = [UIColor clearColor];
    landscape.backgroundColor = [UIColor clearColor];
    //[operation setTitle:todo forState:UIControlStateNormal];
    operation.titleLabel.adjustsFontSizeToFitWidth = TRUE;
}



// Override to allow orientations other than the default portrait
orientation.
-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)
interfaceOrientation {
    // Return YES for supported orientations.
    [self createTextField];
    if(((interfaceOrientation == UIInterfaceOrientationLandscapeLeft)
||
        (interfaceOrientation == UIInterfaceOrientationLand-
scapeRight))) {
        self.view = landscape;
        [operation2 setTitle:todo forState:UIControlStateNormal];
    }
    else if (((interfaceOrientation == UIInterfaceOrientationPor-
trait) ||
              (interfaceOrientation == UIInterfaceOrientationPortrai-
tUpsideDown))) {
        self.view = portrait;
        [operation setTitle:todo forState:UIControlStateNormal];
    }
    NSLog(@"%@",todo);
    return YES;
}


- (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.field2 = nil;
    self.field3 = nil;
    self.portrait = nil;
    self.landscape = nil;
}

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


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

```
    [super dealloc];
}


@end
```

BasisViewController.h

```
#import <UIKit/UIKit.h>
#import "TestAddProBrain2.h"


@interface BasisViewController : UIViewController {

    TestAddProBrain2 *brain;
    NSString *myunion;
    NSString *result;
    NSMutableArray *textFields;
    int noOfEquations;
    IBOutlet UIScrollView *display;
    IBOutlet UIScrollView *display2;
    IBOutlet UIView *portrait;
    IBOutlet UIView *landscape;
    int times;
}

@property (nonatomic,retain) NSString *result;
@property (nonatomic,assign) int noOfEquations;
@property (retain) IBOutlet UIScrollView *display;
@property (retain) IBOutlet UIScrollView *display2;
@property(nonatomic,retain) IBOutlet UIView *portrait;
@property(nonatomic,retain) IBOutlet UIView *landscape;

-(NSString *)checkEmptyTextFields2;
-(IBAction)infoButtonPressed:(UIButton *)sender;
-(IBAction)buttonPressed:(UIButton *)sender;

@end
```

BasisViewController.m

```
#import "BasisViewController.h"
#import "TestAddProBrain2.h"
#import "FurtherView.h"


@implementation BasisViewController

@synthesize noOfEquations;
@synthesize display;
@synthesize display2;
@synthesize result;
@synthesize portrait;
@synthesize landscape;
```

```
-(TestAddProBrain2 *)brain
{
    if (!brain) brain = [[TestAddProBrain2 alloc] init];
    return brain;
}

-(void)showMagic
{
    FurtherView *tapvc = [[FurtherView alloc] init];
    tapvc.title = @"Result";
    [tapvc setResult:self.result];
    [self.navigationController pushViewController:tapvc animat-
ed:YES];
    [tapvc release];
}


-(IBAction)buttonPressed:(UIButton *)sender
{
    NSString *matrix = [self checkEmptyTextFields2];


    if([matrix isEqual:@"Empty"])
    {
        UIAlertView *noInput = [[UIAlertView alloc]
                                initWithTitle:@"Invalid Input"
                                message:@"One of the fields is empty"
                                delegate:self
                                cancelButtonTitle:@"OK"
                                otherButtonTitles:nil];
        [noInput show];
        [noInput release];
    }
    else {
        self.result = [[self brain] computeBasis:matrix];
    }
    [self showMagic];

}

-(void)createTextField
{
    times++;
    textFields = [[NSMutableArray alloc] init];
    [display setContentSize:CGSizeMake(150,noOfEquations*50)];
    [display2 setContentSize:CGSizeMake(150,noOfEquations*50)];

    for(int i=20;i<noOfEquations*50+10;i+=50)
    {
        UITextField *field = [[UITextField alloc] initWith-
Frame:CGRectMake(20, i, 150, 35)];
        field.returnKeyType = UIReturnKeyDone;
        field.placeholder = @"0";
        field.autocapitalizationType = UITextAutocapitalizationType-
Words;
        field.adjustsFontSizeToFitWidth = TRUE;
        field.backgroundColor = [UIColor yellowColor];
        field.keyboardType = UIKeyboardTypeNumbersAndPunctuation;
        field.textAlignment = UITextAlignmentRight;
```

```
        field.contentVerticalAlignment = UIControlContentVerticalA-
lignmentCenter;
        [field addTarget:self
                  action:@selector(textFieldDone:)
        forControlEvents:UIControlEventEditingDidEndOnExit];
        [textFields addObject:field];
        if(((self.interfaceOrientation == UIInterfaceOrientationPor-
trait) ||
            (self.interfaceOrientation == UIInterfaceOrientationPor-
traitUpsideDown))) {

            [display addSubview:field];
            if (times==2) {
                [display2 addSubview:field];
            }
        }
        else if(((self.interfaceOrientation == UIInterfaceOrienta-
tionLandscapeLeft) ||
                 (self.interfaceOrientation == UIInterfaceOrienta-
tionLandscapeRight))) {
            [display2 addSubview:field];
            if (times==2) {
                [display addSubview:field];
            }
        }

        [field release];
    }
    display.scrollEnabled = YES;
    display2.scrollEnabled = YES;

}


-(NSString *)checkEmptyTextFields2
{
    myunion = [[NSString alloc] initWithFormat:@"["];

    for(UITextField *spyros in textFields)
    {
        myunion = [myunion stringByAppendingString:@"["];
        if ([spyros.text length]== 0) {
            return @"Empty";
        }
        else {
            myunion = [myunion stringByAppendingString:spyros.text];
            myunion = [myunion stringByAppendingString:@"],"];
        }
    }
    myunion = [myunion substringToIndex:[myunion length] - 1];
    myunion = [myunion stringByAppendingString:@"]"];

    return myunion;
}

-(IBAction)infoButtonPressed:(UIButton *)sender
{
    UIAlertView *instructionsAlert = [[UIAlertView alloc]
                                      initWithTitle:@"Enter Vectors"
```

```
                                        message:@"Enter each vector in
a textfield, separating their elements by com-
mas.\n\nATTENTION!\n\nYou do NOT have to enter brackets!"
                                        delegate:nil
                                        cancelButtonTitle:@"OK"
                                        otherButtonTitles:nil];
    [instructionsAlert show];
    [instructionsAlert release];
}

- (IBAction)textFieldDone:(id)sender {
    [sender resignFirstResponder];
}

// The designated initializer.  Override if you create the controller
programmatically and want to perform customization that is not appro-
priate for viewDidLoad.
/*
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle
*)nibBundleOrNil {
    self = [super initWithNibName:nibNameOrNil bun-
dle:nibBundleOrNil];
    if (self) {
        // Custom initialization.
    }
    return self;
}
*/


// Implement viewDidLoad to do additional setup after loading the
view, typically from a nib.
- (void)viewDidLoad {
    [super viewDidLoad];
    times=0;
    //self.view.backgroundColor = [UIColor clearColor];

}

// Override to allow orientations other than the default portrait
orientation.
-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)
interfaceOrientation {
    // Return YES for supported orientations.
    [self createTextField];
    if(((interfaceOrientation == UIInterfaceOrientationLandscapeLeft)
||
        (interfaceOrientation == UIInterfaceOrientationLand-
scapeRight))) {
        self.view = landscape;
    }
    else if (((interfaceOrientation == UIInterfaceOrientationPor-
trait) ||
              (interfaceOrientation == UIInterfaceOrientationPortrai-
tUpsideDown))) {
        self.view = portrait;
    }
    self.view.backgroundColor = [UIColor clearColor];
    return YES;
}
```

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


- (void)dealloc {
    [super dealloc];
}


@end
```

TestAddProBrain2.h

```
#import <Foundation/Foundation.h>


@interface TestAddProBrain2 : NSObject {

}

-(NSString *)lala2:(NSString *)operation what:(NSString *)todo;
-(NSString *)lala3:(NSString *)operation sasa:(NSString *)op2 ma-
sa:(NSString *)op3;
-(NSString *)matrix:(NSString *)operation bvec:(NSString *)op2
cvec:(NSString *)op3 operation:(NSString *)op4;
-(NSString *)solveLinear:(NSString *)equations;
-(NSString *)computeBasis:(NSString *)input;

@end
```

TestAddProBrain2.mm

```
#import "TestAddProBrain2.h"
#include <giac/giac.h>
#include <misc.h>
#include <permu.h>


using namespace std;
using namespace giac;

@implementation TestAddProBrain2
```

```
-(NSString *)lala2:(NSString *)operation what:(NSString *)todo
{
    gen e(string([operation UTF8String]),0);
    gen res;


        if([todo isEqual:@"Absolute Value"])
        {
            if(e.type==_VECT){
                res = _abs(e,0);
            }
            else {
                res = gen("-999",0);
            }

        }
        else if([todo isEqual:@"Euclidean Norm"])
        {
            res = _l2norm(e,0);
        }
        else if([todo isEqual:@"Maximum Norm"])
        {
            res = _maxnorm(e,0);
        }
    else if(ckmatrix(e)){  //check if input is a matrix
        if([todo isEqual:@"Rank"])
        {
            /* find the rank of a matrix given */
            res = _rank(e,0);
        }
        else if([todo isEqual:@"LU"])
        {
            if (is_squarematrix(e)) {
                res = lu(e, 0);
            }
            else {
                res = gen("-999",0);
            }

        }
        else if([todo isEqual:@"QR"])
        {
            NSString *myunion = [[NSString alloc] init];

            myunion = [myunion stringByAppendingString:operation];
            myunion = [myunion stringByAppendingString:@",-1"];

            gen e2(string([myunion UTF8String]),0);

            res = qr(e2, 0);
        }
        else if([todo isEqual:@"SVD"])
        {
            res = _svd(e, 0);

        }
        else if([todo isEqual:@"LLL"])
        {
            res = _lll(e, 0);
```

```
        }
        else if([todo isEqual:@"Trace"])
        {
            res = ckmtrace(e, 0);
        }
        else if([todo isEqual:@"Cholesky"])
        {
            if (is_squarematrix(e)) {
                res = _cholesky(e, 0);
            }
            else {
                res = gen("-999",0);
            }
        }
        else if([todo isEqual:@"Transpose Matrix"])
        {
            res = _tran(e,0);
        }
        else if([todo isEqual:@"Image"])
        {
            res = _image(e,0);
        }
        else if([todo isEqual:@"Characteristic Polynomial"])
        {
            res = _pcar(e,0);
        }
        else if([todo isEqual:@"Inverse Matrix"])
        {
            res = _inv(e,0);
        }
        else if([todo isEqual:@"Det"])
        {
            res = _det(e,0);
        }
        else if([todo isEqual:@"Conjugate Transpose Matrix"])
        {
            res = _trn(e,0);
        }
        else if([todo isEqual:@"Gauss - Jordan Reduction"])
        {
            res = _rref(e,0);
        }
        else if([todo isEqual:@"Kernel"])
        {
            res = _ker(e,0);
        }
        else if([todo isEqual:@"Eigenvalues"])
        {
            res = _eigenvals(e,0);
        }
        else if([todo isEqual:@"Eigenvectors"])
        {
            res = _egv(e,0);
        }
    }
    else
    {
        UIAlertView *noInput = [[UIAlertView alloc]
                                initWithTitle:@"Invalid Input"
                                message:@"Invalid Input"
                                delegate:self
```

```
                                cancelButtonTitle:@"OK"
                                otherButtonTitles:nil];
        [noInput show];
        [noInput release];
        res = gen("-99",0);
    }

    res = _map(makevecteur(res,gen("x->approx(x[0])",0),0), 0);

    NSString *test = [[NSString alloc] initWith-
UTF8String:print(res,0).c_str()];
    NSLog(@"%@",test);
    return test;
}

-(NSString *)lala3:(NSString *)operation sasa:(NSString *)op2 ma-
sa:(NSString *)op3
{
    gen e1(string([operation UTF8String]),0);
    gen e2(string([op2 UTF8String]),0);
    gen res;

    if([op3 isEqual:@"Dot Product"])
    {
        res = _dotprod(makevecteur(e1,e2), 0);
    }
    else if([op3 isEqual:@"Cross Product"])
    {
        res = _cross(makevecteur(e1,e2), 0);
    }
    else if([op3 isEqual:@"Multiplication"])
    {
        res = e1*e2;
    }
    else if([op3 isEqual:@"Addition"])
    {
        res = e1+e2;
    }
    else if([op3 isEqual:@"Subtraction"])
    {
        res = e1-e2;
    }
    else if([op3 isEqual:@"Hadamard Product"])
    {
        if((ckmatrix(e1))&&(ckmatrix(e2))){
            res = _hadamard(makevecteur(e1,e2), 0);
        }
    }
    else {
        res = gen("-99",0);
    }


    NSString *test = [[NSString alloc] initWith-
UTF8String:print(res,0).c_str()];
    return test;
}

-(NSString *)matrix:(NSString *)operation bvec:(NSString *)op2
cvec:(NSString *)op3 operation:(NSString *)op4
{
```

```
    gen e1(string([operation UTF8String]),0);

    NSString *myunion = [[NSString alloc] init];

    myunion = [myunion stringByAppendingString:operation];
    myunion = [myunion stringByAppendingString:@","];
    myunion = [myunion stringByAppendingString:op2];
    myunion = [myunion stringByAppendingString:@","];
    myunion = [myunion stringByAppendingString:op3];

    gen e(string([myunion UTF8String]),0);
    gen res;
    if(ckmatrix(e1)){
        if([op4 isEqual:@"Simplex Method"])
        {
            res = _simplex_reduce(e, 0);
        }
        else if([op4 isEqual:@"Pivot"])
        {
            res = _pivot(e,0);
        }
    }
    else {
        res = gen("-99",0);
    }


    NSString *test = [[NSString alloc] initWith-
UTF8String:print(res,0).c_str()];
    return test;
}


-(NSString *)solveLinear:(NSString *)equations;
{
    gen e(string([equations UTF8String]),0);
    gen res = _linsolve(e, 0);

    NSString *test = [[NSString alloc] initWith-
UTF8String:print(res,0).c_str()];
    return test;
}

-(NSString *)computeBasis:(NSString *)input
{
    gen e(string([input UTF8String]),0);
    gen res;
    if(ckmatrix(e))
    {
        res = _basis(e, 0);
    }
    else {
        res = gen("-99",0);
    }

    NSString *test = [[NSString alloc] initWith-
UTF8String:print(res,0).c_str()];
    return test;
}

@end
```