Για μεταγλώτιση ανοίγουμε στο QtCreator το project QtExample και το project Solve . Κάνουμε built to Solve και ύστερα το QtExample και πατάμε play ή ανοίγουμε terminal και τρέχουμε το εκτελέσιμο.

Για να τερματιστεί η είσοδος των σημείων απλά κλείνεις το παράθυρο.

Στα X input / from κ.λ.π ο τερματισμός εισόδου γίνεται με το enter , όπως και όταν εισάγετε d1 και d2!

Σε περιπτωση που αλλάξει η μεταβλητή εμφανίζονται 2 παράθυρα με τις συναρτησεις και τα σημεία ζωγραφισμένα. Απλα επειδή ανοίγουν στην ίδια θέση μετακινείστε το παράθυρο της gnuplot

ΣΤΟΙΧΕΙΑ:

Όνομα	Επίθετο	Α.Μ
Ένρι	Γκάτση	1115200900048
Gerald	Mema	1115200800108

ΟΔΗΓΙΕΣ ΜΕΤΑΓΛΩΤΙΣΗΣ για το κυρίως πρόγραμμα
1)Στο φάκελο QtExample ανοίξτε terminal και πληκτρολογείστε make

ΟΔΗΓΙΕΣ ΜΕΤΑΓΛΩΤΙΣΗΣ για τα tests.
1)make στο φάκελο project_2.6/LibTest

ΟΔΗΓΙΕΣ ΧΡΗΣΗΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

Εκτέλεση : στο φάκελο QtExample μετά το make πληκτρολογείστε ./qtgnuplotlib-example

Μπορείτε να ανοίξεται αρχείο μέσω του File/Open ή πληκτρολογόντας Alt+F. Ύστερα κλικάρετε το Solve και θα εμφανιστούν κατάλληλα μυνήματα στη γραφική διεπαφή.

Κλικάρετε στο Plot equations για να ζωγραφιστούν οι καμπύλες και τα σημεία τομής σε αναδυόμενο παράθυρο της gnuplot.

----------> Προσοχή το πρόγραμμα απαιτεί ο σταθερός όρος να είναι ο πρώτος όρος στην εξίσωση.

Εισαγωγή σημείων :

Κάντε click στο Insert 1st points / 2nd points αντίστοιχα , δώστε τα σημεία που θέλετε και κλείστε το παράθυρο προκειμένου να αποθηκευθούν.

Κάντε click στην επιλογή Solve προκειμένου να δείτε τις λύσεις και με τη Plot equations εμφανίζονται οι καμπύλες και οι ρίζες

Επιπρόσθετα :
Καθάρισμα : make clean

ΟΔΗΓΙΕΣ ΧΡΗΣΗΣ των tests.
Εκτέλεση : ./main
Επιπρόσθετα :

Καθάρισμα : make clean

ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

Δέχεται σαν είσοδο απο αρχείο 2 εξισώσεις . Υπολογίζει τα σημεία τομής βάση της θεωρίας που έχει δωθεί στο μάθημα και μπορεί να αναπαραστήσει τα σημεία και τις καμπύλες σε γραφική διεπαφη.

Διαφορετικά δέχεται σημεία μέσω του ποντικιού στην οθόνη και χρησιμοποιώντας την κλάσση interpolation δημιουργεί τις συναρτήσεις που περνάνε απο αυτα τα σημεία. Υπολογίζει τις τομές εαν υπάρχουν και δίνει επιλογή για γραφική απεικόνιση.

Bonus: το πρόγραμμα λειτουργεί και για πραγματικούς συντελεστές.

Επίλυση συστημάτων μη γραμμικών εξισώσεων:

Η άσκηση αυτή επικεντρώνεται στο να λύνει ένα πρόβλημα ιδιοτιμών-ιδιοδιανυσμάτων με το κλασσικό (στάνταρ) τρόπο ή με το γενικευμένο.

Ροή εκτέλεσης του προγράμματος:

Εισαγωγή απο αρχείο :

Visualization.cpp:

1) Κλήση συνάρτησης on_actionOpen_triggered() .

2) Το πρόγραμμα περιμένει να πατηθεί το πλήκτρο Solve προκειμένου να υπολογιστούν οι ρίζες.

3) Η Solve() καλεί την handle_read_from_file(Handler & handler).

4) Κάλειτε η Handler::read(char * filename) . Αυτή διαβάζει τις συναρτήσεις απο το αρχείο που δόθηκε και φτιάχνεις πίνακες συντελεστών.

5) Παράγεται ο Sylvester , το πολυώνυμο μητρώων και καλείται η συνάρτηση Handler::Solve()

Handler::Solve() :

Βρίσκει το βαθμό της κρυμμένης μεταβλητής.

Ελέγχει εαν είναι αντιστρέψιμος ο πίνακας μέσω του κάππα.

Σε περίπτωση που δεν είναι αντιστρέψιμος . Ελέγχει μήπως μπορεί να βρεί καλύτερο κάππα.

Εαν βρήκε το flag allaksaMetavlhth θα γίνει 1 το πρόβλημα λύνεται με το generalized. Σε αυτή τη περίπτωση οι λύσεις φυλάσονται στο αρχείο points2.txt . 'Υστερα αλλάζουν οι πίνακες συντελεστών και το πρόβλημα λύνεται με τη standard μέθοδο , οι λύσεις φυλάσονται στο αρχείο points.txt κατα την αποικόνιση με τη gnuplot ανοίγουν 2 παράθυρα. 1 εμφανίζει τις λύσεις που βρήκα με το generalized και το άλλο εμφανίζει τις λύσεις που βρήκαμε με το standard.

Αλλίως λύνεται με την αντίστοιχη μέθοδο εαν είναι αντιστρέψιμος ο πίνακας ή όχι.

Περίπτωση generalized λύσης:

Καλείται η συνάρτηση GeneralProblem::solveWithGeneralProblem(...)

Αυτή κοιτάζει το βαθμό της κρυμμένης μεταβλητήτης και παράγει τα κατάλληλα L0 , L1 διατηρώντας του πίνακες lambda και V.

Έπειτα καλείται η συνάρτηση Handler::generate_rest_solution(...)

Εδώ για κάθε τιμή του lambda ελέγχεται εαν μηδενίζονται τα πολυώνυμα. Στη περίπτωση που έχω πολλαπλότητα στη κρυμμένη μεταβλητή καλείται η συνάρτηση Handler::calc_x_with_multiplicity(...).

Αυτή λύνει το πρόβλημα με τα βήματα που δίνονται στη θεωρία.

Περίπτωση standard προβήματος:

Καλείται η Companion::solveWithInvert(...) η οποία ανάλογα με το βαθμό της κρυμμένης μεταβλητής παράγει τον αντίστοιχο Companion και τα lambda και V.

Ύστερα καλείται η Handler::generate_rest_solution(...)

Εισαγωγή με σημεία :

Visualization.cpp:

1) Κλήση συνάρτησης on_Insert_points_clicked() .

1.1) Όταν κλικάρονται σημεία καλείται η συνάρτηση eventFilter(...) . Στη οποία γίνεται η αποθήκευση των σημείων που εισάγει ο χρήστης στα αρχεία points.txt και points2.txt (για το 2 σετ σημείων).

2) Κλήση συνάρτησης on_Insert_2nd_points_clicked() . Και κλήση του 1.1)

3)Κλήση συνάρτηση on_Solve_clicked() όταν πληκτρολογηθεί το κουμπί Solve απο τον χρήστη.

4) Κλήση συνάρτησης handle_read_points(...)

5)Κλήση Handler::handle_Points(...)

Handler::handle_Points(...) :

Παράγεται ο Interpolation Matrix και γίνεται συγγραφή των παραγώμενων εξισώσεων στο αρχείο generated_functions.txt το οποίο χειρίζεται ο handler

καλείται η Handler::read() , παράγεται ο Sylvester και το πολυώνυμο μητρώων και εκτελείται η Handler::solve().

Αρχεία .cpp :

Στη βιβλιοθήκη Solve στο κατάλογο project_2.6:
Ολα τα .cpp αρχεια που βρίσκονται εκει και στους υποκαταλόγους

Στο κατάλογο QtExample αναπτύχθηκε το αρχείο visualization.cpp

Αρχεία .h/hpp :

Στη βιβλιοθήκη Solve στο κατάλογο project_2.6:
Ολα τα .h/.hpp αρχεια που βρίσκονται εκει και στους υποκαταλόγους

Στο κατάλογο QtExample αναπτύχθηκε το αρχείο visualization.h