ΕΛΑΧΙΣΤΟΠΟΙΗΣΗ ΚΥΡΤΗΣ ΣΥΝΑΡΤΗΣΗΣ ΜΙΑΣ ΜΕΤΑΒΛΗΤΗΣ ΣΕ ΔΟΣΜΕΝΟ ΔΙΑΣΤΗΜΑ

Γεώργιος Γκιουλέκας – ΑΕΜ 10636

ΤΕΧΝΙΚΕΣ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ Άσκηση 1η

Περιεχόμενα

Εισαγωγή

Στον φάκελο που βρίσκεται αυτή η αναφορά υπάρχει ένας φάκελος Project με τον κώδικα του project και ένας φάκελος Figures με τα διαγράμματα που υπάρχουν και στην παρούσα αναφορά, χωρισμένα σε επιμέρους φακέλους ανά θέμα. Στον φάκελο του project υπάρχει ένα αρχείο **main.m** το οποίο κάνει αρχικοποίηση των συναρτήσεων και του αρχικού διαστήματος. Μέσα από αυτό το αρχείο μπορεί κανείς να τρέξει το αντίστοιχο section για κάθε θέμα ξεχωριστά. Εφόσον χρειαστεί, μπορεί να εκτελεστεί το αντίστοιχο section για κάθε υποερώτημα ενός θέματος, με την προϋπόθεση **να έχει εκτελεστεί το πρώτο section αρχικοποίησης στο main.m** . Έπειτα υπάρχουν 4 αρχεία .m με την υλοποίηση του κάθε αλγορίθμου καθώς και άλλα 4 αρχεία .m με τον κώδικα για το κάθε θέμα (για την δημιουργία των plots). Τα αρχεία task_*.m ουσιαστικά καλούν τις αντίστοιχες μεθόδους και αναπαριστούν γραφικά τα αποτελέσματά τους.

Τέλος, όλα τα .m αρχεία έχουν αναλυτικές εξηγήσεις σε σχόλια, για το τι κάνει ο κώδικας σε κάθε γραμμή, πώς λειτουργεί κάθε συνάρτηση, τι ορίσματα δέχεται, τι επιστρέφει και πολλές άλλες πληροφορίες.

Το πρόβλημα

Θέλουμε να ελαχιστοποιήσουμε 3 κυρτές συναρτήσεις $f(x)$, $x \in [a, b]$

Έχουμε το αρχικό διάστημα αναζήτησης [-1, 3] και τις συναρτήσεις προς ελαχιστοποίηση (αντικειμενικές):

$f_1(x) = (x-2)^2 + x \cdot \ln(x+3)$

$f_2(x) = e^{-2x} + (x-2)^2$

$f_3(x) = e^x \cdot (x^3 - 1) + (x-1) \cdot \sin x$

Για σκοπούς επαλήθευσης βρίσκουμε τα σημεία ελαχίστου της κάθε συνάρτησης στο δοθέν διάστημα με χρήση του γραφικού λογισμικού Desmos:

Για την f_1: 1.14991

Για την f_2: 2.01767

Για την f_3: 0.52008

Μέθοδοι αναζήτησης ελαχίστου χωρίς χρήση παραγώγων

Οι μέθοδοι που θα περιγράψουμε είναι επαναληπτικοί αλγόριθμοι που βρίσκουν με προκαθορισμένη ακρίβεια l > 0 το σημείο ελαχίστου μιας μονοδιάστατης αυστηρά σχεδόν-κυρτής συνάρτησης f(x) (αντικειμενική συνάρτηση), δοθέντος ενός αρχικού διαστήματος αναζήτησης $[a_1, b_1]$ το οποίο περιέχει το σημείο ελαχίστου της f. Η λειτουργία των αλγορίθμων αυτών βασίζεται στην διαδοχική σμίκρυνση του διαστήματος αναζήτησης έως ότου το αυτό να έχει εύρος μικρότερο της επιθυμητής ακρίβειας. Τότε ο αλγόριθμος τερματίζει και γνωρίζουμε πως το σημείο ελαχίστου βρίσκεται εντός του τελικού διαστήματος αναζήτησης.

Το βασικό θεώρημα

Η μείωση του διαστήματος αναζήτησης σε κάθε επανάληψη βασίζεται σε ένα θεώρημα σύμφωνα με το οποίο για δύο εσωτερικά σημεία του

$$x_1, x_2 \in [a, b] \; \tau\acute{\epsilon}\tau o\iota\alpha \; \acute{\omega}\sigma\tau\epsilon \; x_1 < x_2.$$

$$A\nu \; f(x_1) < f(x_2) \; \tau\acute{o}\tau\epsilon \; f(x) \geq f(x_1) \; \forall \; x \in (x_2, b].$$

$$A\nu \; f(x_1) \geq f(x_2) \; \tau\acute{o}\tau\epsilon \; f(x) \geq f(x_2) \; \forall \; x \in [a, x_1)$$

Οπότε μπορούμε σε κάθε επανάληψη να συγκρίνουμε την τιμή της συνάρτησης f στα δύο σημεία x_1 και x_2 και να περιορίζουμε το διάστημα αναζήτησης σε $[x_1, b]$ ή $[a, x_2]$ ανάλογα με το αποτέλεσμα της σύγκρισης.

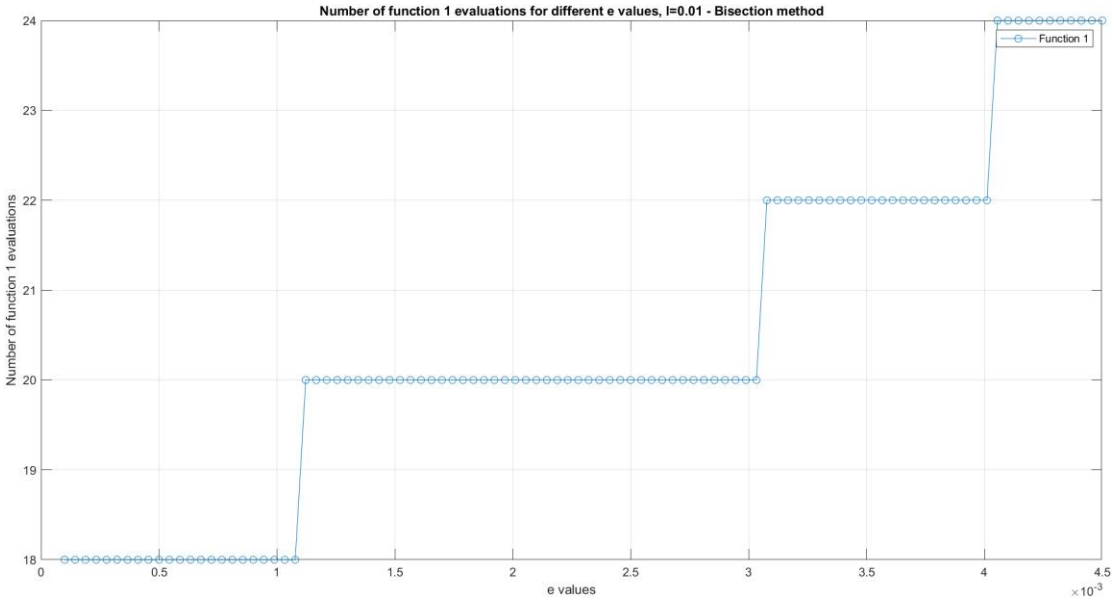
Μέθοδος της διχοτόμου

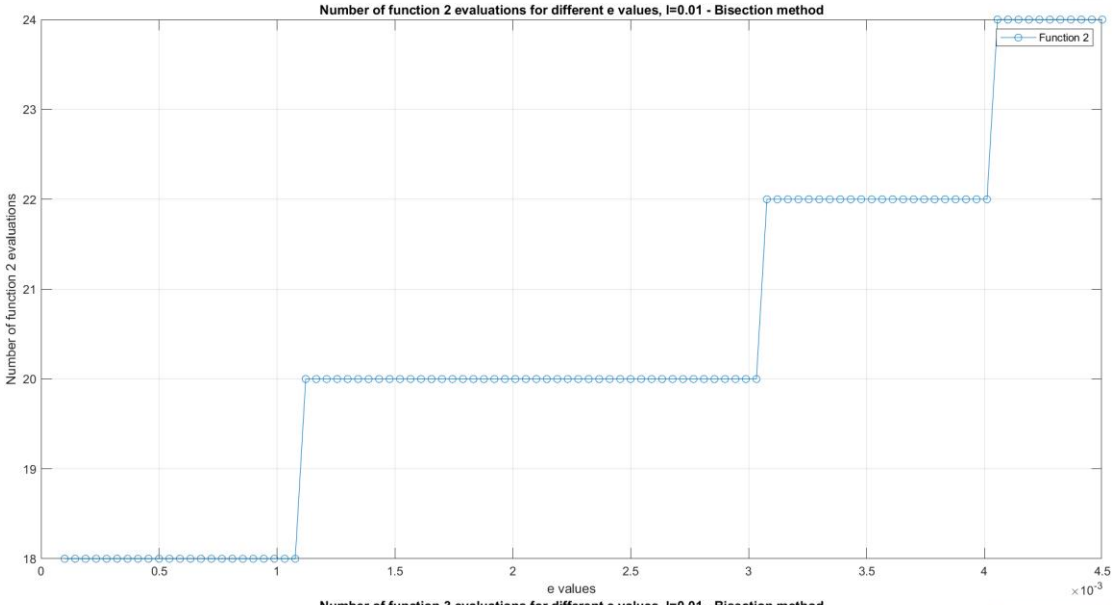
Στην μέθοδο της διχοτόμου σε κάθε επανάληψη το διάστημα αναζήτησης χωρίζεται σε τρία ανισομερή υπό-διαστήματα με τη βοήθεια δύο εσωτερικών σημείων συμμετρικά τοποθετημένων σε απόσταση ε από το μέσο του διαστήματος. Εφαρμόζουμε το βασικό θεώρημα σε αυτά τα δύο σημεία και περιορίζουμε κατάλληλα το διάστημα αναζήτησης. Η απόσταση από την διχοτόμο (μέσο) του διαστήματος προσδιορίζεται από την επιθυμητή ακρίβεια με την οποία θέλουμε να βρούμε το σημείο ελαχίστου. Επειδή τα δύο βοηθητικά σημεία είναι εσωτερικά του διαστήματος πρέπει να προσέξουμε να ισχύει ε < l/2 αλλιώς το διάστημα θα σταματήσει να περιορίζεται.

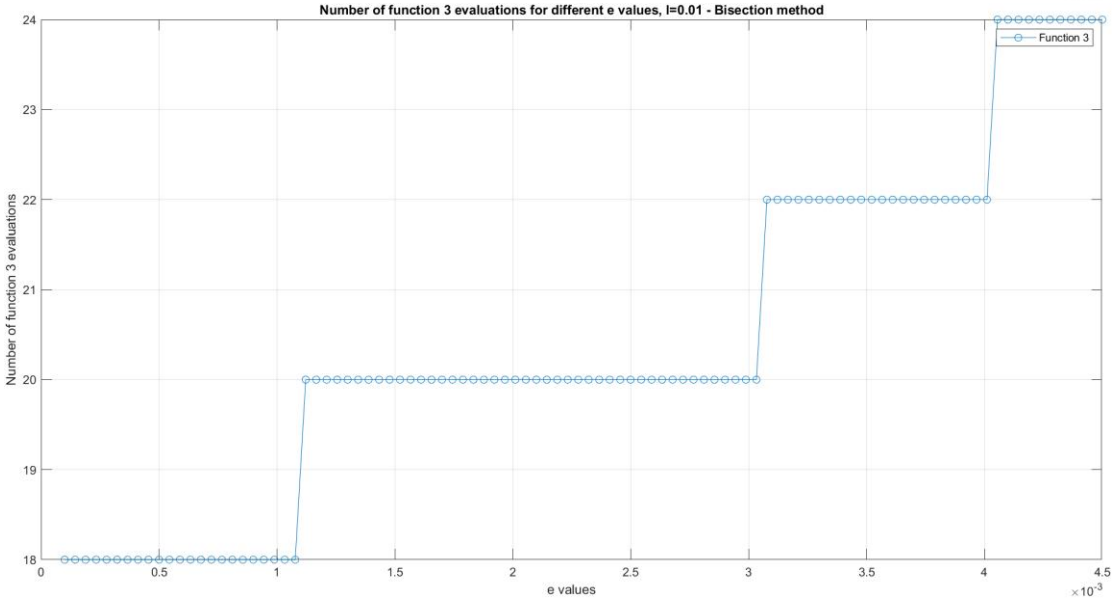
Υλοποιούμε την μέθοδο της διχοτόμου στο Matlab και την εφαρμόζουμε σε 3 διαφορετικά προβλήματα:

Υποπρόβλημα 1: Σταθερό εύρος αναζήτησης l=0.01, μεταβαλλόμενη απόσταση από την διχοτόμο ε

Θέλουμε να μελετήσουμε την μεταβολή των υπολογισμών της αντικειμενικής συνάρτησης ως προς την μεταβολή της απόστασης από την διχοτόμο ε>0, καθώς διατηρούμε σταθερή την ακρίβεια του τελικού διαστήματος l=0.01. Τρέχουμε τον κώδικα στο Matlab και φτιάχνουμε τις γραφικές παραστάσεις για 100 τιμές του ε ομοιόμορφα κατανεμημένες από 0.0001 ως 0.0045 (θέλουμε όπως είπαμε ε<l/2 = 0.005) με την εντολή linspace(0.0001, 0.0045, 100). Παρατηρούμε πως οι γραφικές απεικονίσεις για κάθε συνάρτηση επικαλύπτονται (πράγμα λογικό αφού ο αριθμός υπολογισμών δεν εξαρτάται με κάποιον τρόπο από την αντικειμενική συνάρτηση, μόνο από το αρχικό διάστημα αναζήτησης και την επιθυμητή ακρίβεια l) οπότε κάνουμε ξεχωριστό διάγραμμα για κάθε συνάρτηση ώστε να φαίνονται καθαρά οι παραστάσεις τους. Παρατηρούμε μια κλιμακωτή αύξηση του αριθμού υπολογισμών της αντικειμενικής συνάρτησης όσο αυξάνεται η απόσταση από την διχοτόμο ε. Αυτό συμβαίνει επειδή αυξάνοντας την απόσταση από την διχοτόμο ουσιαστικά μειώνεται η απόσταση των εσωτερικών σημείων από τα άκρα του διαστήματος αναζήτησης, οπότε σε κάθε επανάληψη το διάστημα περιορίζεται λιγότερο και άρα χρειαζόμαστε περισσότερες επαναλήψεις (και υπολογισμούς της f) ώστε να επιτύχουμε την επιθυμητή ακρίβεια l.

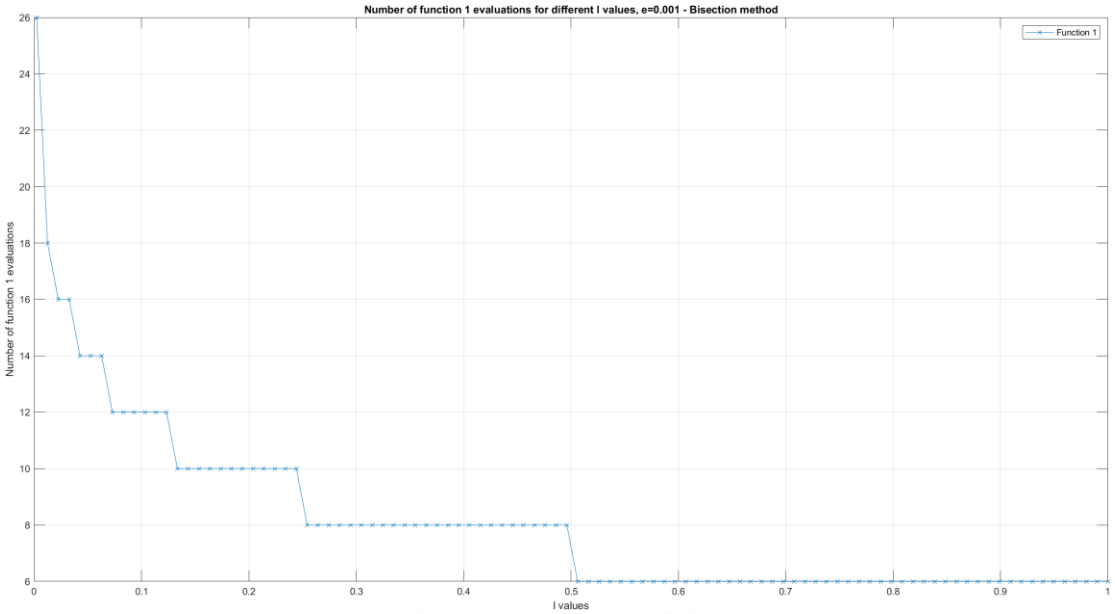

Number of function 1 evaluations for different e values, l=0.01 - Bisection method
Function 1
Number of function 1 evaluations
e values
×10^-3

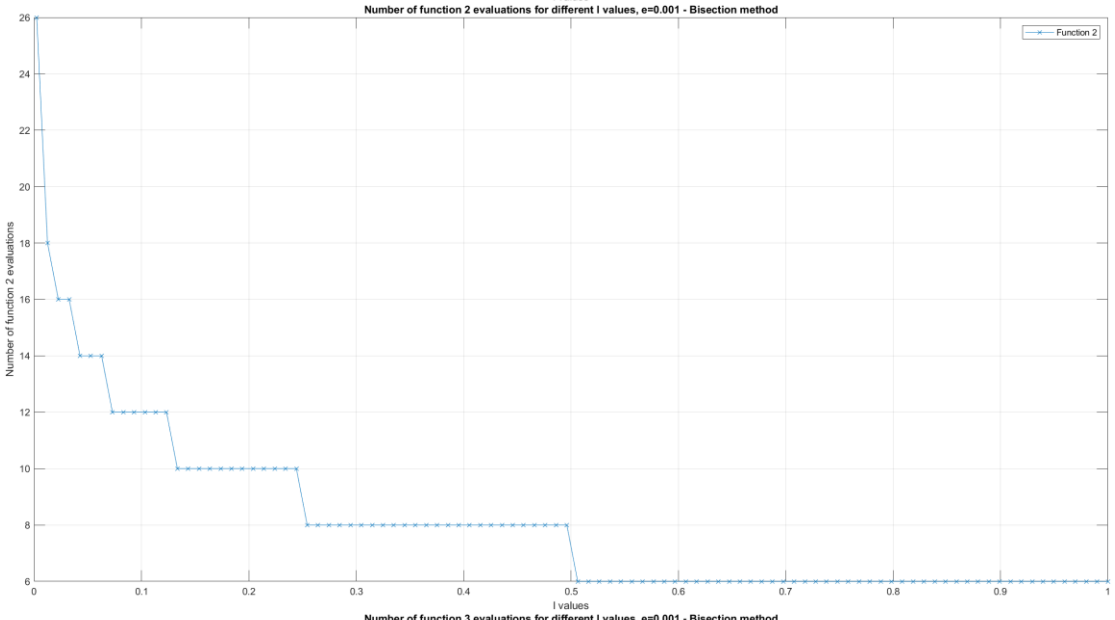

Number of function 2 evaluations for different e values, l=0.01 - Bisection method
Function 2
Number of function 2 evaluations
e values
×10^-3

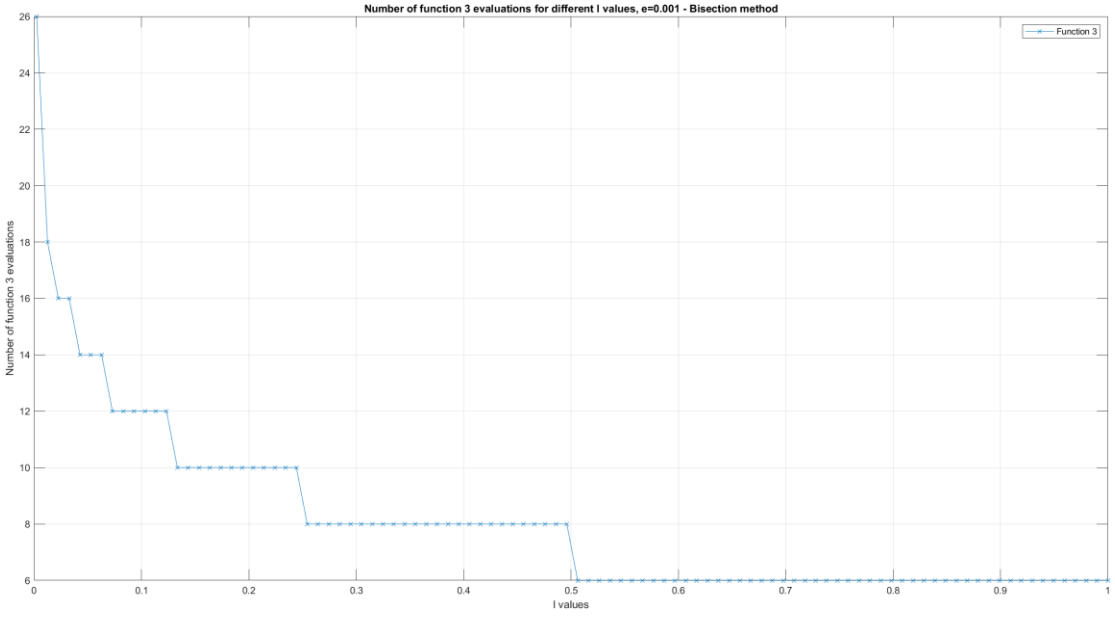

Number of function 3 evaluations for different e values, l=0.01 - Bisection method
Function 3
Number of function 3 evaluations
e values
×10^-3

Υποπρόβλημα 2: Σταθερή απόσταση από την διχοτόμο ε=0.001, μεταβαλλόμενο εύρος αναζήτησης l

Τώρα θα κάνουμε το αντίθετο. Θα διατηρήσουμε σταθερή την απόσταση των εσωτερικών σημείων από την διχοτόμο σε κάθε επανάληψη ε=0.001 και θα μεταβάλλουμε την επιθυμητή ακρίβεια του τελικού διαστήματος αναζήτησης l. Τρέχουμε τον κώδικα στο Matlab και φτιάχνουμε τις γραφικές παραστάσεις για 100 τιμές του l ομοιόμορφα κατανεμημένες από 0.0025 ως 1 (θέλουμε $l > 2\varepsilon = 0.002$) με την εντολή linspace(0.0025, 1, 100). Παρατηρούμε πως πάλι για κάθε συνάρτηση οι γραφικές παραστάσεις επικαλύπτονται οπότε κάνουμε ένα διάγραμμα για κάθε συνάρτηση ώστε να φαίνονται καλύτερα. Φαίνεται πως ο αλγόριθμος είναι υπολογιστικά «ακριβός» για μικρότερες τιμές της επιθυμητής ακρίβειας του τελικού διαστήματος, μάλιστα ο αριθμός υπολογισμών αυξάνεται πολύ γρήγορα όσο μειώνουμε το εύρος του. Αυτό είναι αναμενόμενο καθώς για μεγαλύτερη ακρίβεια απαιτούμε περισσότερες επαναλήψεις.

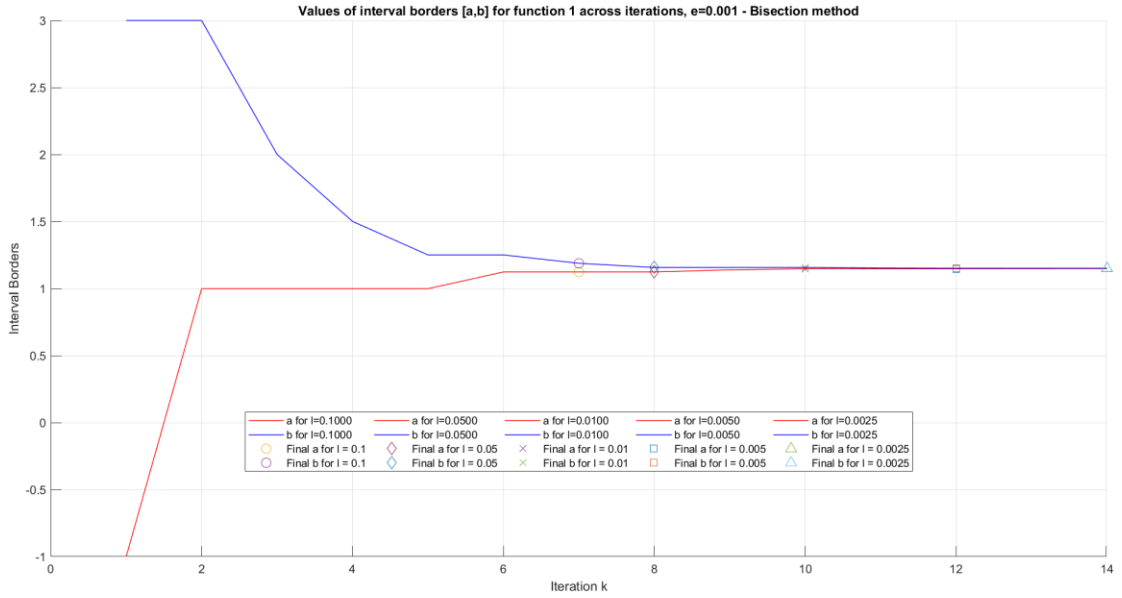

Number of function 1 evaluations for different l values, e=0.001 - Bisection method
Function 1
Number of function 1 evaluations
l values

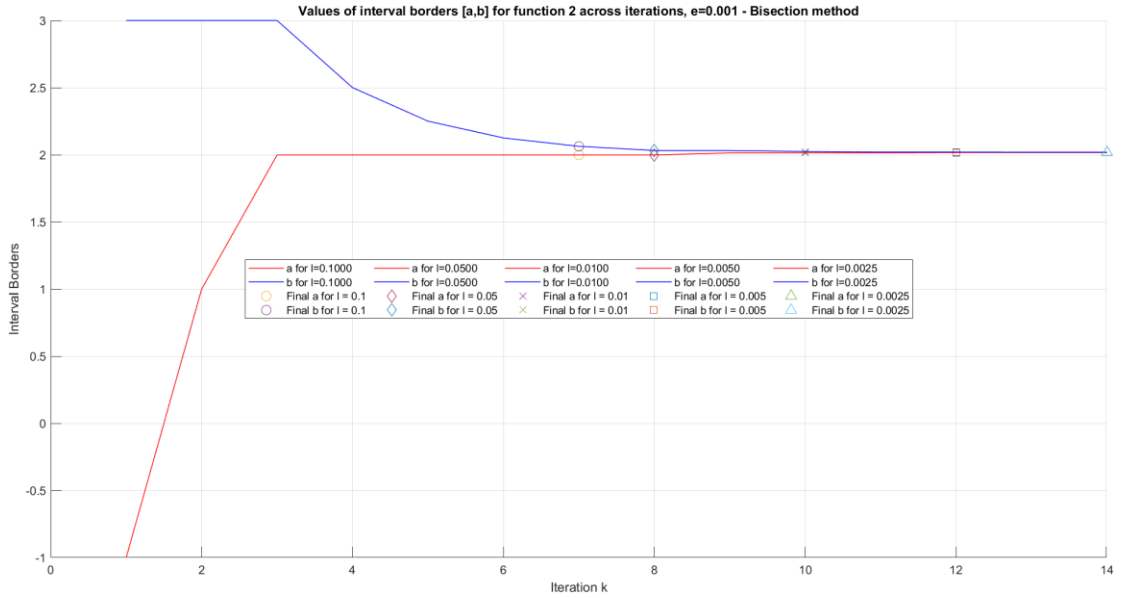

Number of function 2 evaluations for different l values, e=0.001 - Bisection method
Function 2
Number of function 2 evaluations
l values

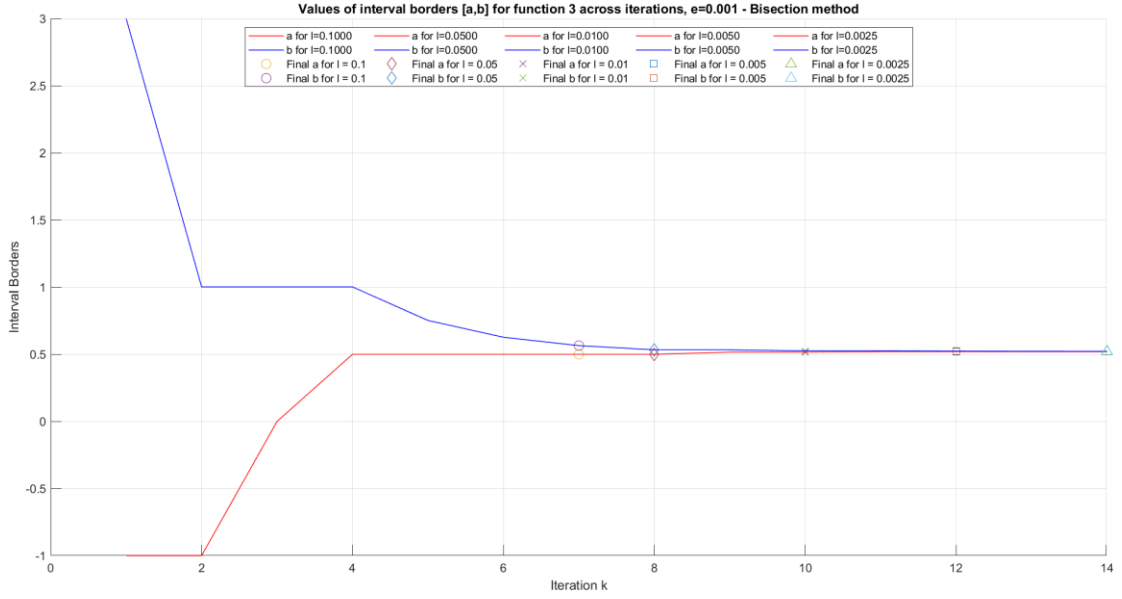

Number of function 3 evaluations for different l values, e=0.001 - Bisection method
Function 3
Number of function 3 evaluations
l values

Υποπρόβλημα 3: Μεταβολή των άκρων του διαστήματος αναζήτησης σε κάθε επανάληψη, για διάφορες τιμές τελικής ακρίβειας l

Τώρα θα κρατήσουμε την απόσταση από την διχοτόμο σταθερή ε=0.001 και επιλέγοντας μερικές τιμές για την τελική ακρίβεια l = [0.1, 0.05, 0.01, 0.005, 0.0025] θα τρέξουμε τον αλγόριθμο. Θα κάνουμε 3 διαγράμματα, ένα για κάθε συνάρτηση, όπου θα φαίνονται τα άκρα του διαστήματος αναζήτησης σε κάθε επανάληψη και θα φροντίσουμε να σημαδέψουμε τα τελικά διαστήματα για καθεμία από τις προκαθορισμένες τιμές του l. Παρατηρούμε ότι τα διαστήματα αναζήτησης συγκλίνουν όντως στα σημεία ελαχίστων που βρήκαμε στην αρχή, προφανώς με μεγαλύτερη ακρίβεια για περισσότερες επαναλήψεις. Επίσης παρατηρούμε πως το πλήθος των επαναλήψεων ως τον τερματισμό εξαρτάται από την απαιτούμενη ακρίβεια εντοπισμού του ελαχίστου. Τέλος όπως φαίνεται από τα διαγράμματα οι κόκκινες και μπλε γραμμές επικαλύπτονται, αφού για δύο διαφορετικές τιμές του l τα διαγράμματα συγκλίνουν με τον ίδιο τρόπο μέχρι να τερματίσει ο αλγόριθμος για το μεγαλύτερο l και μετά το διάστημα συγκλίνει ακόμη περισσότερο για μεγαλύτερο l.


Values of interval borders [a,b] for function 1 across iterations, e=0.001 - Bisection method
Interval Borders
Iteration k
a for l=0.1000
a for l=0.0500
a for l=0.0100
a for l=0.0050
a for l=0.0025
b for l=0.1000
b for l=0.0500
b for l=0.0100
b for l=0.0050
b for l=0.0025
Final a for l = 0.1
Final a for l = 0.05
Final a for l = 0.01
Final a for l = 0.005
Final a for l = 0.0025
Final b for l = 0.1
Final b for l = 0.05
Final b for l = 0.01
Final b for l = 0.005
Final b for l = 0.0025


Values of interval borders [a,b] for function 2 across iterations, e=0.001 - Bisection method
Interval Borders
Iteration k
a for l=0.1000
a for l=0.0500
a for l=0.0100
a for l=0.0050
a for l=0.0025
b for l=0.1000
b for l=0.0500
b for l=0.0100
b for l=0.0050
b for l=0.0025
Final a for l = 0.1
Final a for l = 0.05
Final a for l = 0.01
Final a for l = 0.005
Final a for l = 0.0025
Final b for l = 0.1
Final b for l = 0.05
Final b for l = 0.01
Final b for l = 0.005
Final b for l = 0.0025


Values of interval borders [a,b] for function 3 across iterations, e=0.001 - Bisection method
Interval Borders
Iteration k
a for l=0.1000
a for l=0.0500
a for l=0.0100
a for l=0.0050
a for l=0.0025
b for l=0.1000
b for l=0.0500
b for l=0.0100
b for l=0.0050
b for l=0.0025
Final a for l = 0.1
Final a for l = 0.05
Final a for l = 0.01
Final a for l = 0.005
Final a for l = 0.0025
Final b for l = 0.1
Final b for l = 0.05
Final b for l = 0.01
Final b for l = 0.005
Final b for l = 0.0025

Μέθοδος του Χρυσού Τομέα

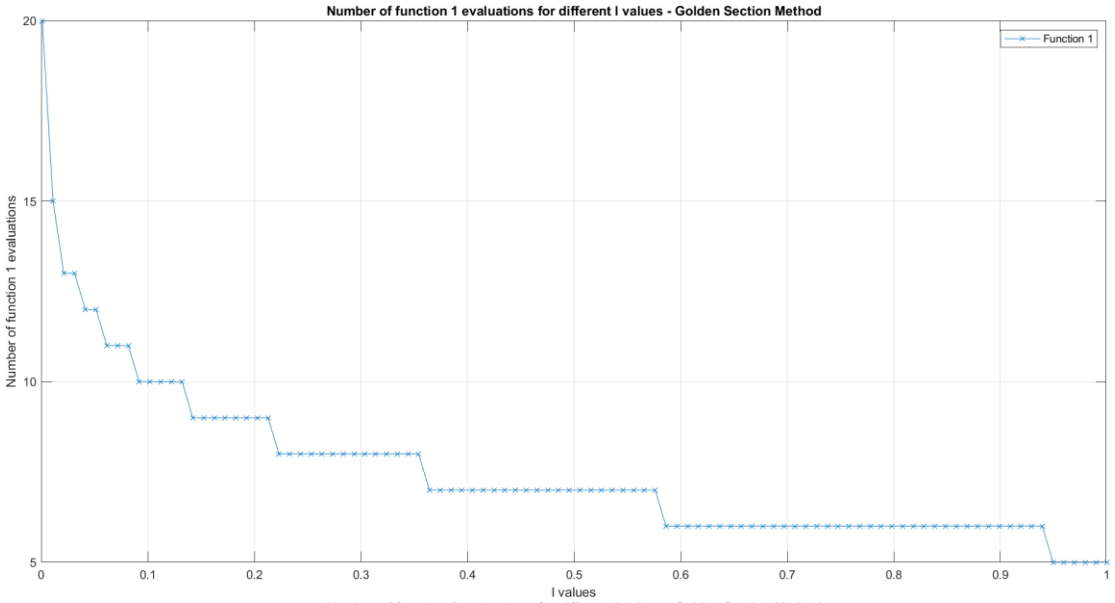
Η ιδιαιτερότητα της μεθόδου του Χρυσού Τομέα είναι ότι ο λόγος του μειωμένου διαστήματος προς το αρχικό σε κάθε επανάληψη είναι σταθερός και ίσος με $\gamma \cong 0.618$. Το γ είναι ο αντίστροφος του αριθμού φ, γνωστού και ως «χρυσή τομή» και από εκεί πήρε ο αλγόριθμος το όνομά του. Εφαρμόζοντας το βασικό θεώρημα στις αποτιμήσεις της αντικειμενικής συνάρτησης στα δύο εσωτερικά σημεία σε κάθε επανάληψη μπορούμε να μειώνουμε το διάστημα αναζήτησης. Στην μέθοδο αυτή τα εσωτερικά σημεία διαλέγονται ώστε να ικανοποιείται η αναλογία μείωσης του διαστήματος καθώς και ένα από τα δύο νέα εσωτερικά σημεία να ταυτίζεται με ένα από τα δύο σημεία της προηγούμενης επανάληψης. Κατά αυτόν τον τρόπο χρειαζόμαστε δύο υπολογισμούς της αντικειμενικής συνάρτησης στην αρχή, ενώ μετά σε κάθε επανάληψη μόνο έναν νέο υπολογισμό αφού έχουμε μόνο ένα νέο εσωτερικό σημείο.

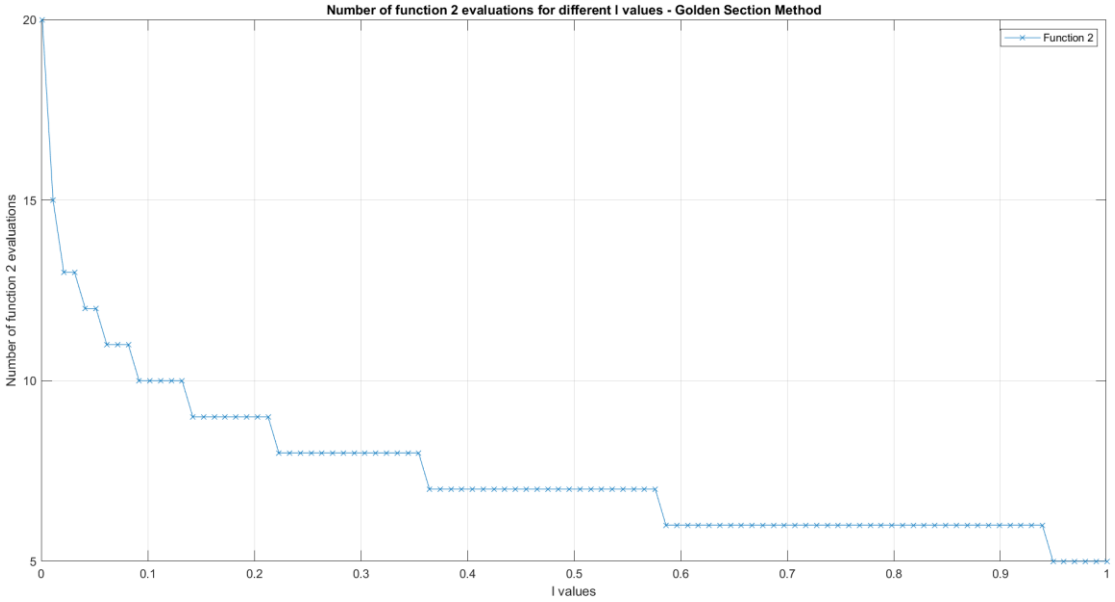
Υλοποιούμε τον αλγόριθμο του Χρυσού Τομέα στο Matlab και τον εφαρμόζουμε για 2 προβλήματα:

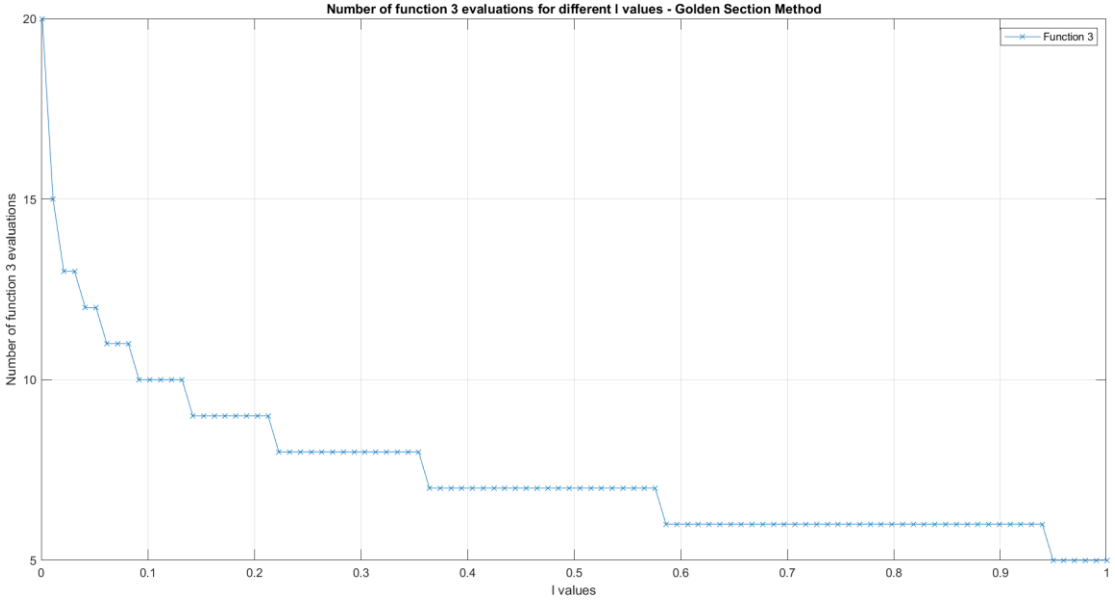
Υποπρόβλημα 1: Πλήθος υπολογισμών αντικειμενικής συνάρτησης με μεταβλητό τελικό εύρος αναζήτησης l

Τρέχουμε τον κώδικα στο Matlab για 100 τιμές του l ομοιόμορφα κατανεμημένες από 0.001 έως 1 με την εντολή linspace(0.001, 1, 100). Κάνουμε 3 διαφορετικές γραφικές παραστάσεις, μία για κάθε συνάρτηση ώστε να μην επικαλύπτονται και όπως πριν διαπιστώνουμε πως ο αριθμός υπολογισμών της αντικειμενικής συνάρτησης μειώνεται δραματικά για μεγαλύτερο επιθυμητό εύρος τελικού διαστήματος, δηλαδή μικρότερη ακρίβεια στον προσδιορισμό του ελαχίστου.

Μια παρατήρηση σχετικά με την υλοποίηση του αλγορίθμου: στο βιβλίο στην σελίδα 111 όπου περιγράφει τον αλγόριθμο, στα βήματα 2 και 3 βλέπουμε πως στην k-οστή επανάληψη (όπου επανάληψη θεωρούμε την είσοδο στα βήματα 2 ή 3) υπολογίζει την συνάρτηση στο σημείο $x_{i,k+1}, i = 1$ ή 2 χωρίς να γνωρίζει εάν για k=k+1 θα ικανοποιείται η συνθήκη του βήματος 1 $\beta\kappa - \alpha\kappa < l$. Έτσι, στην υλοποίηση του βιβλίου ο αλγόριθμος θα κάνει μια επιπλέον άχρηστη αποτίμηση της αντικειμενικής συνάρτησης στην τελευταία επανάληψη η οποία δεν θα χρησιμεύσει σε κάποια μετέπειτα σύγκριση, αφού ο αλγόριθμος θα τερματίσει στον αμέσως επόμενο έλεγχο. Στην σελίδα 114 βρίσκει τον αναμενόμενο αριθμό υπολογισμών της αντικειμενικής συνάρτησης f τον οποίον ονομάζει n. Σύμφωνα με αυτόν τον τύπο προκύπτει ένας λιγότερος υπολογισμός της f, ουσιαστικά δεν λαμβάνει υπόψιν τον έξτρα υπολογισμό που ανέφερα παραπάνω. Θα μπορούσα να αποφύγω αυτόν τον επιπλέον υπολογισμό αν στα βήματα 2 και 3 έκανα έλεγχο των a_{k+1} $\kappa\alpha\iota$ β_{k+1} αφού τα ενημερώσω, για να ξέρω αν θα μπω στην επόμενη επανάληψη, πριν κάνω τον υπολογισμό της f. Όμως όπως είπα ακολούθησα την υλοποίηση του βιβλίου, οπότε βγάζω έναν παραπάνω υπολογισμό από ότι θα έπρεπε σύμφωνα με τον τύπο της σελίδας 114.

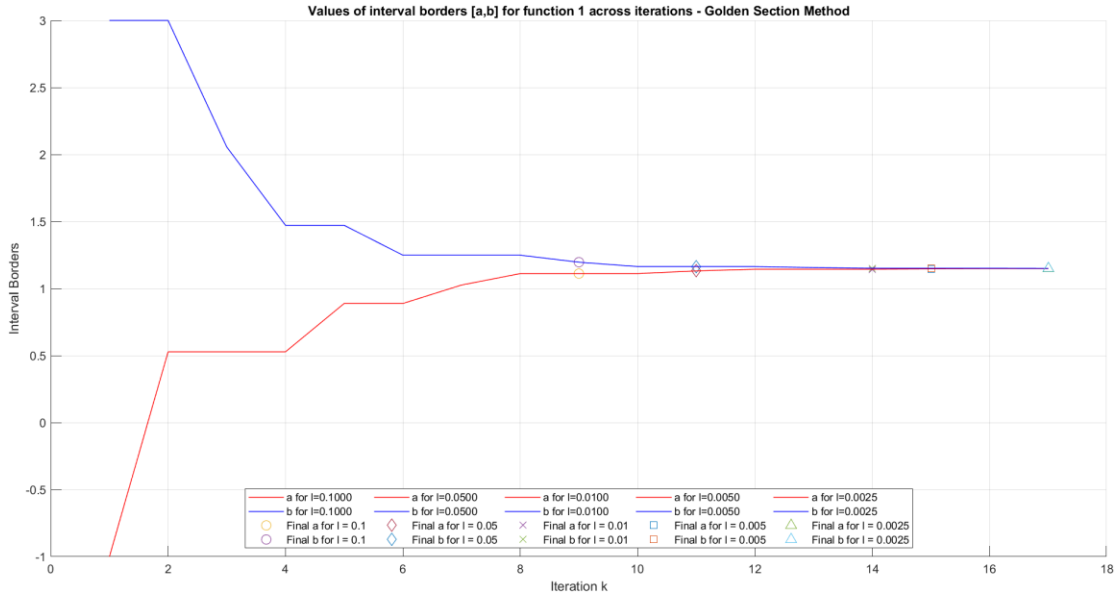

Number of function 1 evaluations for different l values - Golden Section Method
Function 1
Number of function 1 evaluations
l values
20
15
10
5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

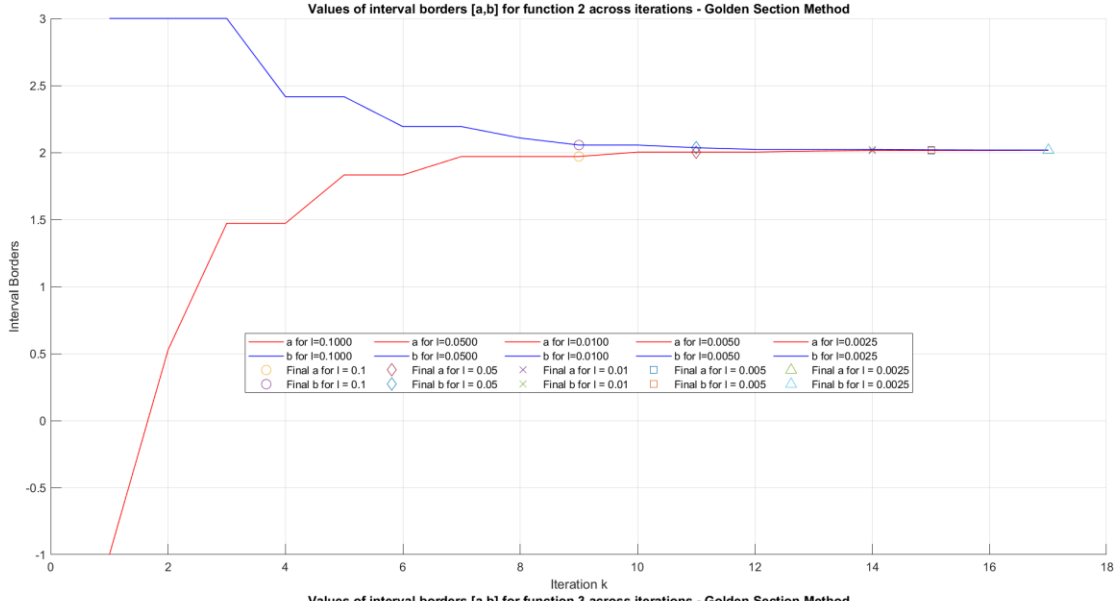

Number of function 2 evaluations for different l values - Golden Section Method
Function 2
Number of function 2 evaluations
l values
20
15
10
5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

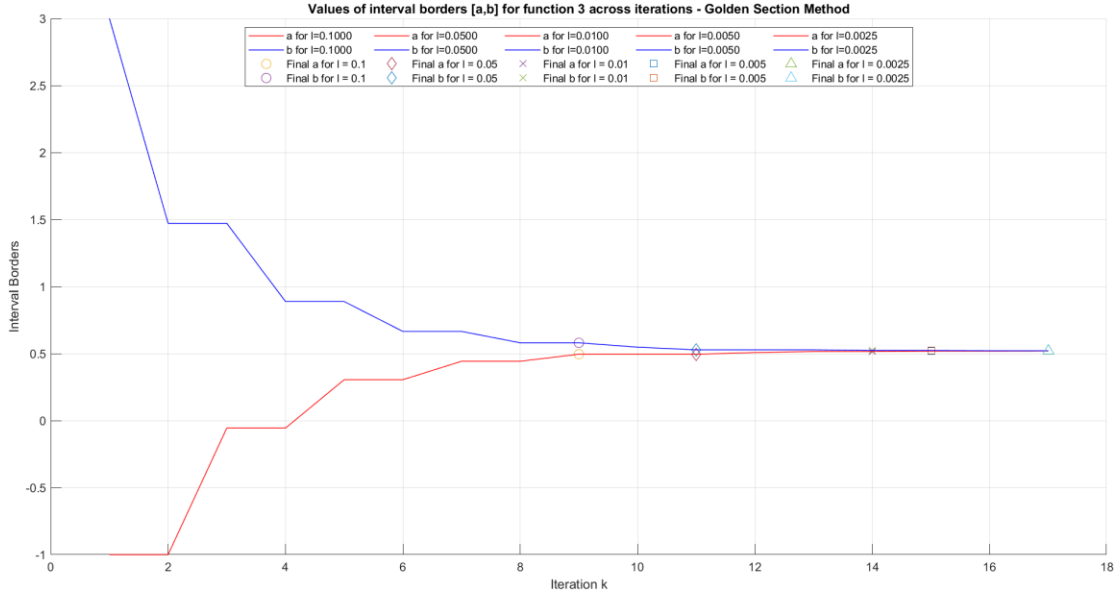

Number of function 3 evaluations for different l values - Golden Section Method
Function 3
Number of function 3 evaluations
l values
20
15
10
5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Υποπρόβλημα 2: Μεταβολή των άκρων του διαστήματος αναζήτησης σε κάθε επανάληψη, για διάφορες τιμές τελικής ακρίβειας l

Για τις ίδιες τιμές του l που χρησιμοποίησα στην μέθοδο της διχοτόμου κάνω τα διαγράμματα για κάθε συνάρτηση των ορίων του διαστήματος αναζήτησης ως προς τον αριθμό επαναλήψεων. Παρατηρώ πως όντως τα διαστήματα συγκλίνουν προς το πραγματικό σημείο ελαχίστου όσο αυξάνονται οι επαναλήψεις.


Values of interval borders [a,b] for function 1 across iterations - Golden Section Method
Interval Borders
Iteration k
a for l=0.1000
a for l=0.0500
a for l=0.0100
a for l=0.0050
a for l=0.0025
b for l=0.1000
b for l=0.0500
b for l=0.0100
b for l=0.0050
b for l=0.0025
Final a for l = 0.1
Final a for l = 0.05
Final a for l = 0.01
Final a for l = 0.005
Final a for l = 0.0025
Final b for l = 0.1
Final b for l = 0.05
Final b for l = 0.01
Final b for l = 0.005
Final b for l = 0.0025


Values of interval borders [a,b] for function 2 across iterations - Golden Section Method
Interval Borders
Iteration k
a for l=0.1000
a for l=0.0500
a for l=0.0100
a for l=0.0050
a for l=0.0025
b for l=0.1000
b for l=0.0500
b for l=0.0100
b for l=0.0050
b for l=0.0025
Final a for l = 0.1
Final a for l = 0.05
Final a for l = 0.01
Final a for l = 0.005
Final a for l = 0.0025
Final b for l = 0.1
Final b for l = 0.05
Final b for l = 0.01
Final b for l = 0.005
Final b for l = 0.0025


Values of interval borders [a,b] for function 3 across iterations - Golden Section Method
Interval Borders
Iteration k
a for l=0.1000
a for l=0.0500
a for l=0.0100
a for l=0.0050
a for l=0.0025
b for l=0.1000
b for l=0.0500
b for l=0.0100
b for l=0.0050
b for l=0.0025
Final a for l = 0.1
Final a for l = 0.05
Final a for l = 0.01
Final a for l = 0.005
Final a for l = 0.0025
Final b for l = 0.1
Final b for l = 0.05
Final b for l = 0.01
Final b for l = 0.005
Final b for l = 0.0025

Μέθοδος Fibonacci

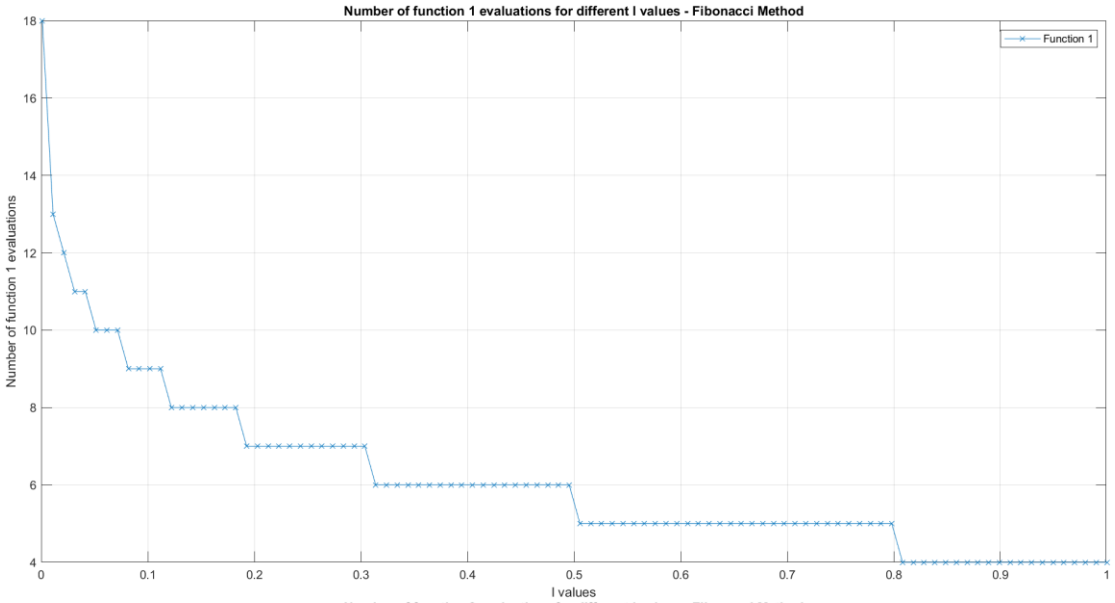
Η μέθοδος Fibonacci μοιάζει με αυτήν του Χρυσού Τομέα. Η ιδιαιτερότητά της είναι ότι απαιτεί τον εκ των προτέρων προσδιορισμό των συνολικών αποτιμήσεων της αντικειμενικής συνάρτησης. Ο προσδιορισμός αυτός γίνεται αξιοποιώντας την ακολουθία Fibonacci $F_0 = F_1 = 1,\ F_n = F_{n-1} + F_{n-2}, n \geq 2$

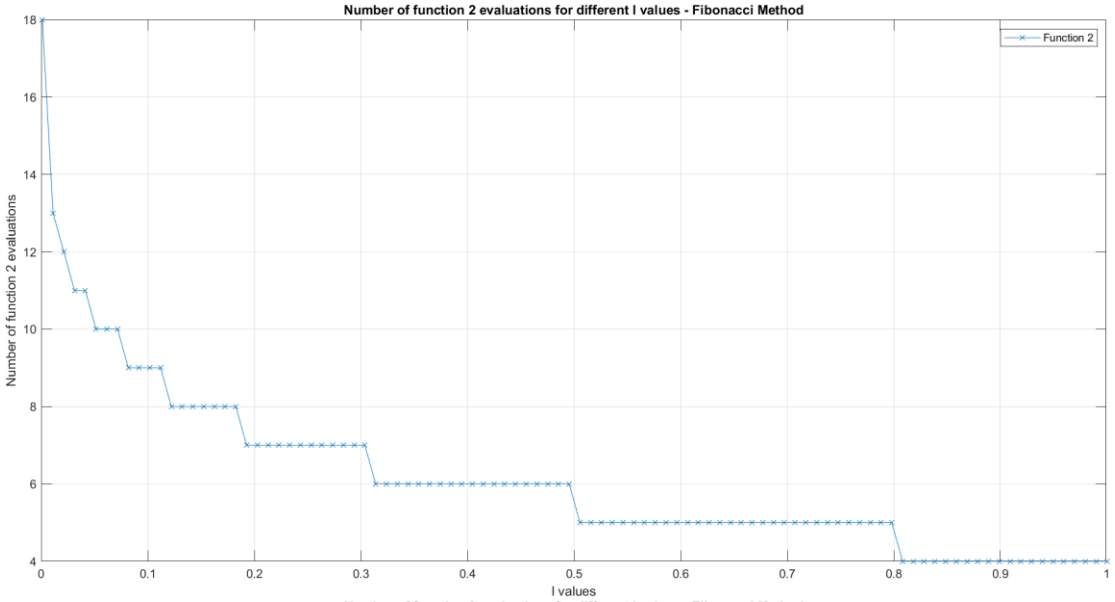
Παρατήρηση: Επειδή στο Matlab οι δείκτες στους πίνακες ξεκινάνε από 1 και όχι από 0, κάθε φορά που αναφέρομαι στον πίνακα με τους αριθμούς Fibonacci χρησιμοποιώ δείκτη αυξημένο κατά 1. Επίσης όταν υπολογίζω τον αριθμό των υπολογισμών της f ψάχνοντας τον πρώτο αριθμό Fibonacci που είναι μεγαλύτερος από $\frac{\beta-\alpha}{l}$ αφαιρώ 1 από τον δείκτη του αριθμού που βρήκα για τον ίδιο λόγο που ανέφερα πριν (πχ ο 4ος Fibonacci είναι ο 5ος που βρίσκω επειδή ξεκινάω την αρίθμηση από το 1 κι όχι το 0).

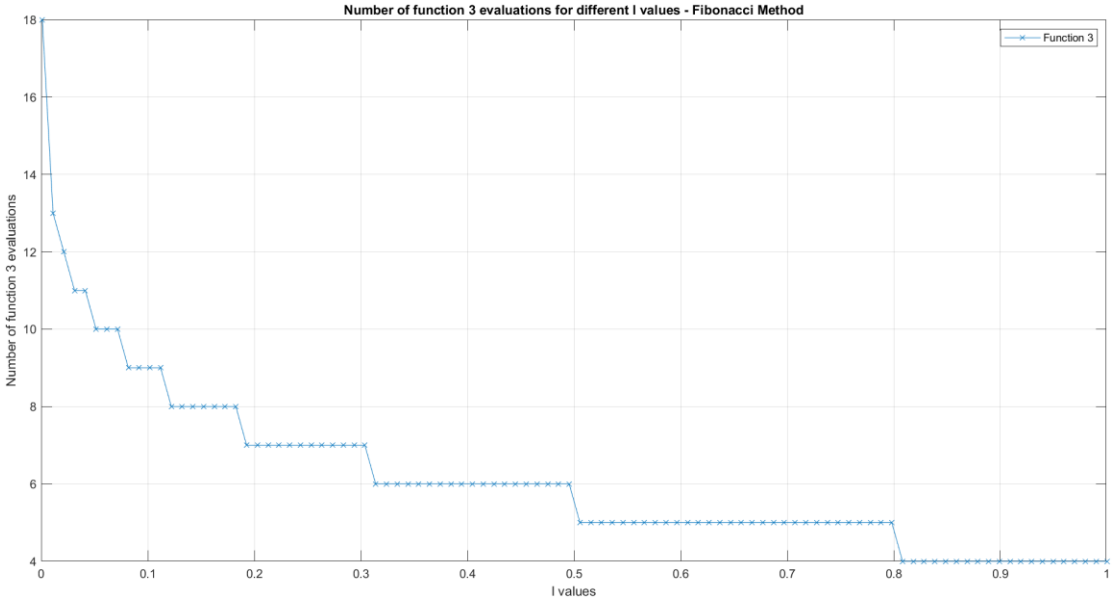
Υλοποιούμε τον αλγόριθμο Fibonacci στο Matlab και τον εφαρμόζουμε στα παρακάτω προβλήματα:

Υποπρόβλημα 1: Πλήθος υπολογισμών αντικειμενικής συνάρτησης με μεταβλητό τελικό εύρος αναζήτησης l

Για αντίστοιχες τιμές του l με το προηγούμενο θέμα πάλι παρατηρώ αντίστοιχα αποτελέσματα: ραγδαία μείωση των υπολογισμών της αντικειμενικής συνάρτησης όσο μειώνεται το ζητούμενο εύρος του τελικού διαστήματος.

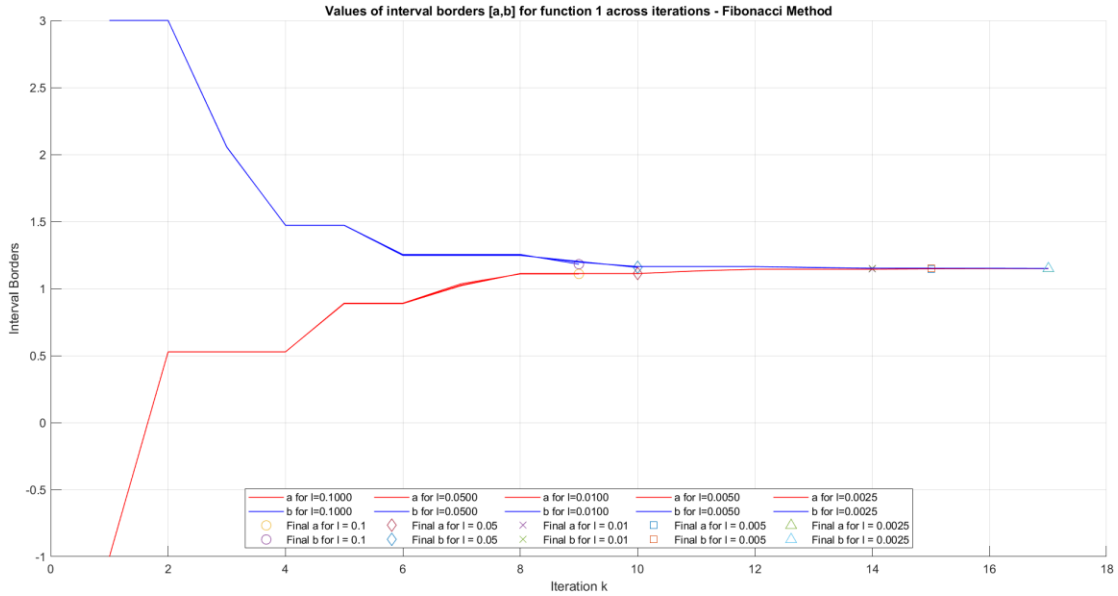

Number of function 1 evaluations for different l values - Fibonacci Method
Function 1
Number of function 1 evaluations
l values

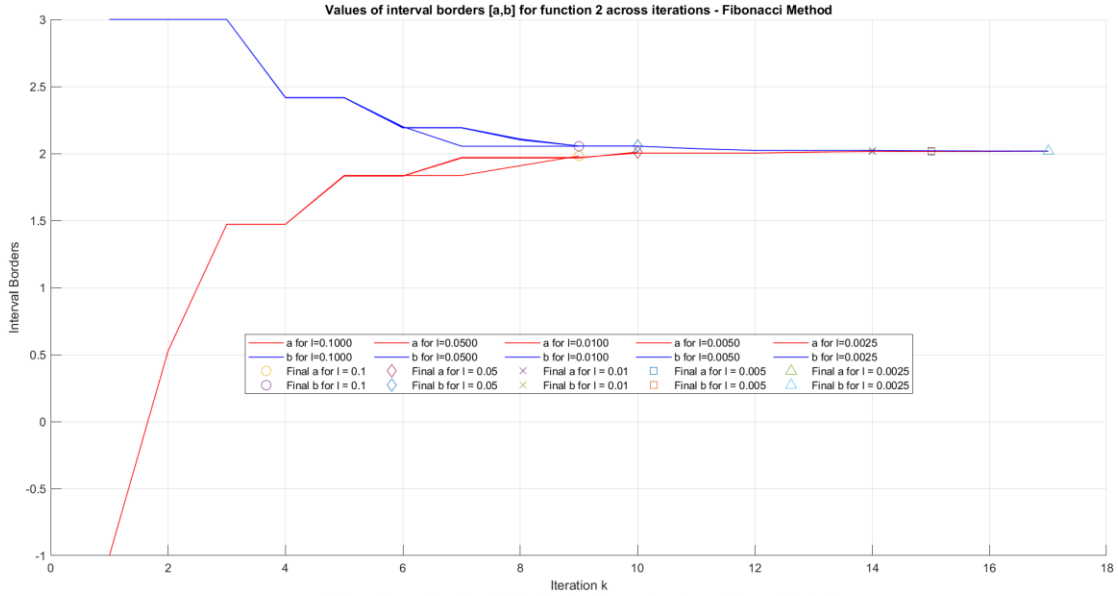

Number of function 2 evaluations for different l values - Fibonacci Method
Function 2
Number of function 2 evaluations
l values

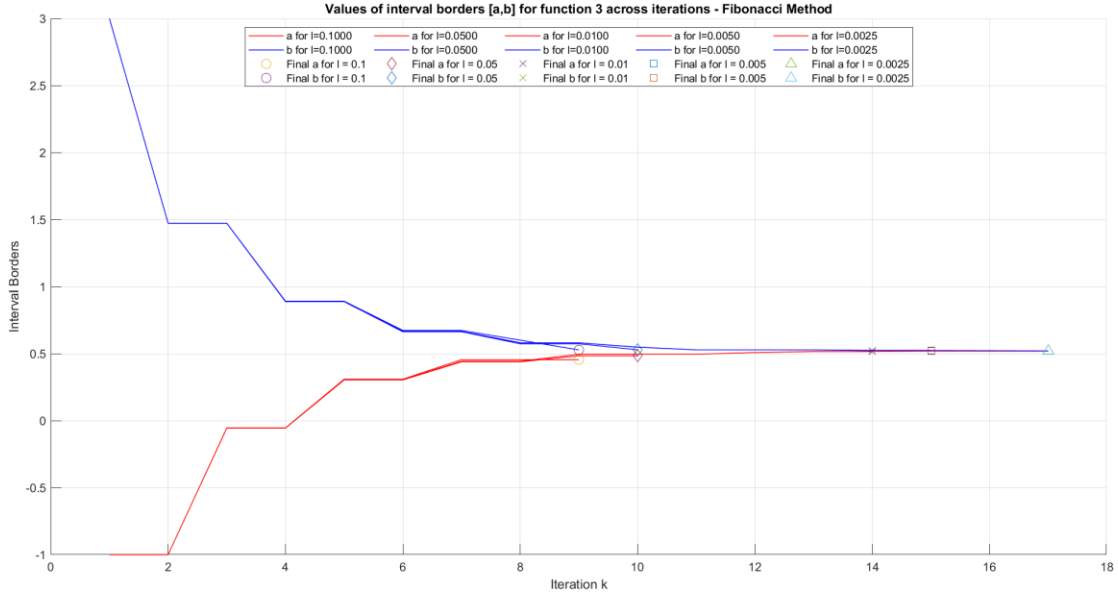

Number of function 3 evaluations for different l values - Fibonacci Method
Function 3
Number of function 3 evaluations
l values

Υποπρόβλημα 2: Μεταβολή των άκρων του διαστήματος αναζήτησης σε κάθε επανάληψη, για διάφορες τιμές τελικής ακρίβειας l

Για τις ίδιες τιμές του l που χρησιμοποίησα πριν κάνω τα διαγράμματα για κάθε συνάρτηση των ορίων του διαστήματος αναζήτησης ως προς τον αριθμό επαναλήψεων. Παρατηρώ πως όντως τα διαστήματα συγκλίνουν προς το πραγματικό σημείο ελαχίστου όσο αυξάνονται οι επαναλήψεις. Ωστόσο μια διαφορά με τις προηγούμενες μεθόδους είναι πως στην μέθοδο Fibonacci το διάστημα για διαφορετικές τιμές του l συγκλίνει διαφορετικά στο ελάχιστο, για αυτό και οι γραφικές παραστάσεις για διαφορετικές τιμές του l δεν επικαλύπτονται όπως πριν. Αυτό συμβαίνει επειδή το πού θα τοποθετηθούν τα εσωτερικά σημεία στην k-οστή επανάληψη (και άρα τα νέα όρια του διαστήματος) εξαρτάται από τον λόγο δύο αριθμών Fibonacci οι οποίοι είναι διαφορετικοί για κάθε τιμή του λόγου $\frac{\beta-\alpha}{l}$. Εφόσον το l καθορίζει το n, τότε και ο λόγος των αριθμών Fibonacci εξαρτάται από το n δεδομένου ότι βρισκόμαστε στην k-οστή επανάληψη. Οπότε για διαφορετικές τιμές του l τα εσωτερικά σημεία (και άρα τα όρια του διαστήματος) είναι διαφορετικά για σταθερό k, και έτσι εξηγείται η απόκλιση των μπλε και κόκκινων γραμμών στα γραφήματα.


Values of interval borders [a,b] for function 1 across iterations - Fibonacci Method
Interval Borders
Iteration k
a for l=0.1000
a for l=0.0500
a for l=0.0100
a for l=0.0050
a for l=0.0025
b for l=0.1000
b for l=0.0500
b for l=0.0100
b for l=0.0050
b for l=0.0025
Final a for l = 0.1
Final a for l = 0.05
Final a for l = 0.01
Final a for l = 0.005
Final a for l = 0.0025
Final b for l = 0.1
Final b for l = 0.05
Final b for l = 0.01
Final b for l = 0.005
Final b for l = 0.0025


Values of interval borders [a,b] for function 2 across iterations - Fibonacci Method
Interval Borders
Iteration k
a for l=0.1000
a for l=0.0500
a for l=0.0100
a for l=0.0050
a for l=0.0025
b for l=0.1000
b for l=0.0500
b for l=0.0100
b for l=0.0050
b for l=0.0025
Final a for l = 0.1
Final a for l = 0.05
Final a for l = 0.01
Final a for l = 0.005
Final a for l = 0.0025
Final b for l = 0.1
Final b for l = 0.05
Final b for l = 0.01
Final b for l = 0.005
Final b for l = 0.0025


Values of interval borders [a,b] for function 3 across iterations - Fibonacci Method
Interval Borders
Iteration k
a for l=0.1000
a for l=0.0500
a for l=0.0100
a for l=0.0050
a for l=0.0025
b for l=0.1000
b for l=0.0500
b for l=0.0100
b for l=0.0050
b for l=0.0025
Final a for l = 0.1
Final a for l = 0.05
Final a for l = 0.01
Final a for l = 0.005
Final a for l = 0.0025
Final b for l = 0.1
Final b for l = 0.05
Final b for l = 0.01
Final b for l = 0.005
Final b for l = 0.0025

Μέθοδοι με χρήση παραγώγων

Μέθοδος της διχοτόμου με χρήση παραγώγων

Έστω μια ψευδοκυρτή και άρα διαφορίσιμη συνάρτηση f(x) σε κλειστό διάστημα $[a, b]$. Έστω ότι στην k-οστή επανάληψη το διάστημα αναζήτησης είναι $[a\kappa, b\kappa]$. Έστω επίσης ότι η παράγωγος της f στο σημείο $x_k \in [a, b]$, $\frac{df(x_k)}{dx}$ είναι γνωστή. Τότε:

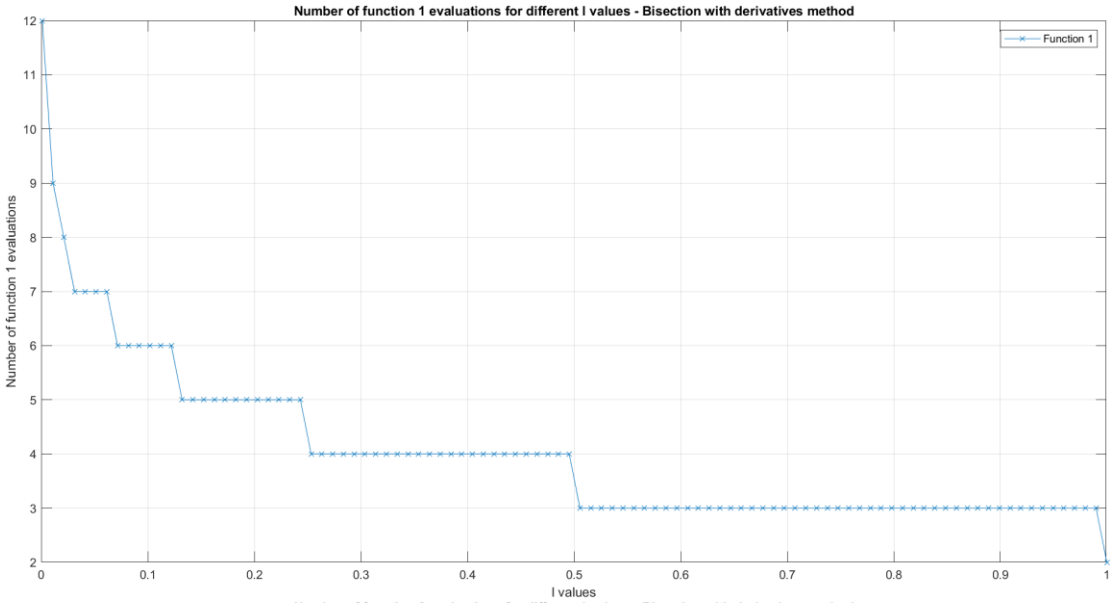
1. Αν $\frac{df(x_k)}{dx} = 0$, το σημείο x_k είναι σημείο ελαχίστου λόγω ψευδοκυρτότητας
2. Αν $\frac{df(x_k)}{dx} > 0$, τότε για $x > x_k$ είναι $(x - x_k) \cdot \frac{df(x_k)}{dx} > 0$ άρα λόγω ψευδοκυρτότητας $f(x) \geq f(x_k)$. Επομένως το νέο διάστημα αναζήτησης είναι $[a_k, x_k]$.
3. Αν $\frac{df(x_k)}{dx} < 0$, τότε για $x < x_k$ είναι $(x - x_k) \cdot \frac{df(x_k)}{dx} > 0$ άρα λόγω ψευδοκυρτότητας $f(x) \geq f(x_k)$. Επομένως το νέο διάστημα αναζήτησης είναι $[x_k, b_k]$.

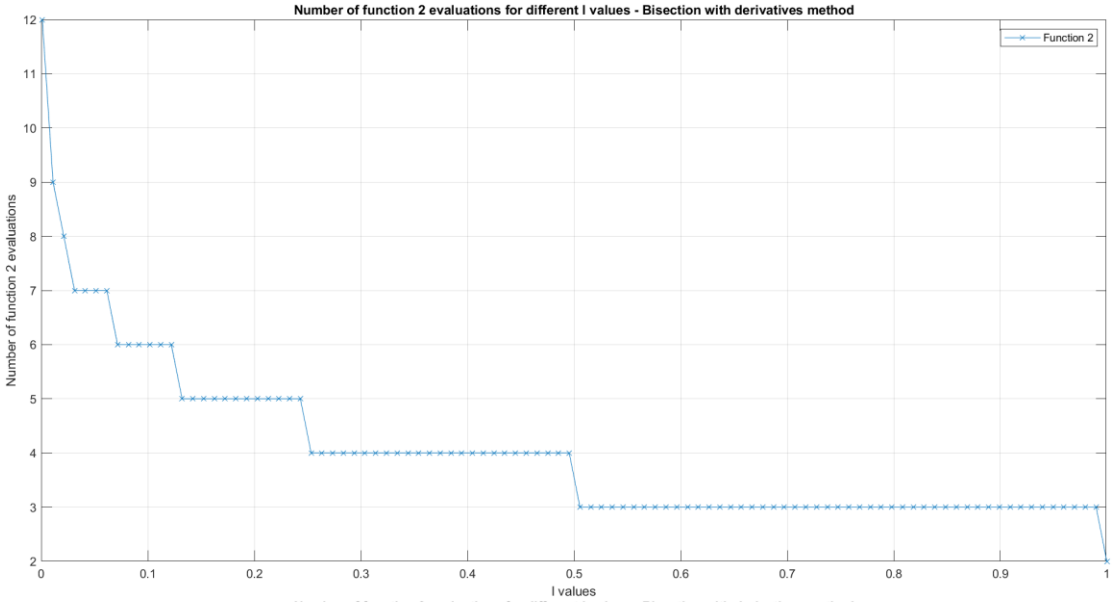
Το σημείο x_k προσδιορίζεται ώστε να ελαχιστοποιείται το εύρος του διαστήματος αναζήτησης οπότε η βέλτιστη τοποθέτησή του είναι στο $\frac{a_k + b_k}{2}$ (ουσιαστικά προσομοιάζει έναν αλγόριθμο binary search).

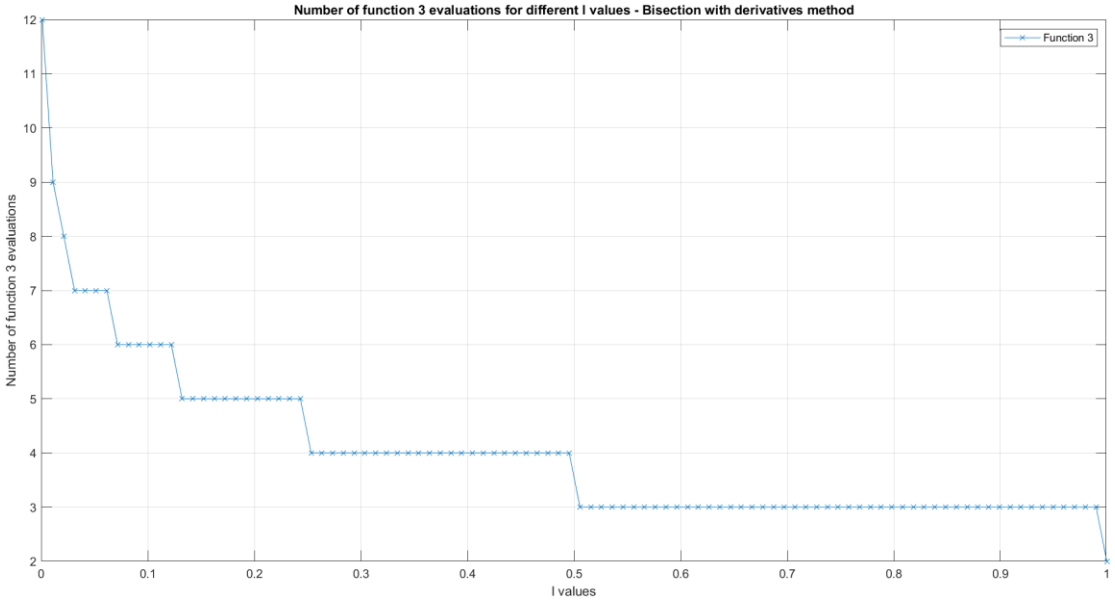
Υλοποιούμε τον αλγόριθμο στο Matlab και τον εφαρμόζουμε στα γνωστά προβλήματα:

Υποπρόβλημα 1: Πλήθος υπολογισμών της παραγώγου της αντικειμενικής συνάρτησης με μεταβλητό τελικό εύρος αναζήτησης l

Πλέον μετράμε τον αριθμό των υπολογισμών της παραγώγου της αντικειμενικής συνάρτησης για κάποιες τιμές του l. Παίρνουμε τις ίδιες τιμές του l με τα προηγούμενα θέματα και κάνουμε τα διαγράμματα για κάθε συνάρτηση:

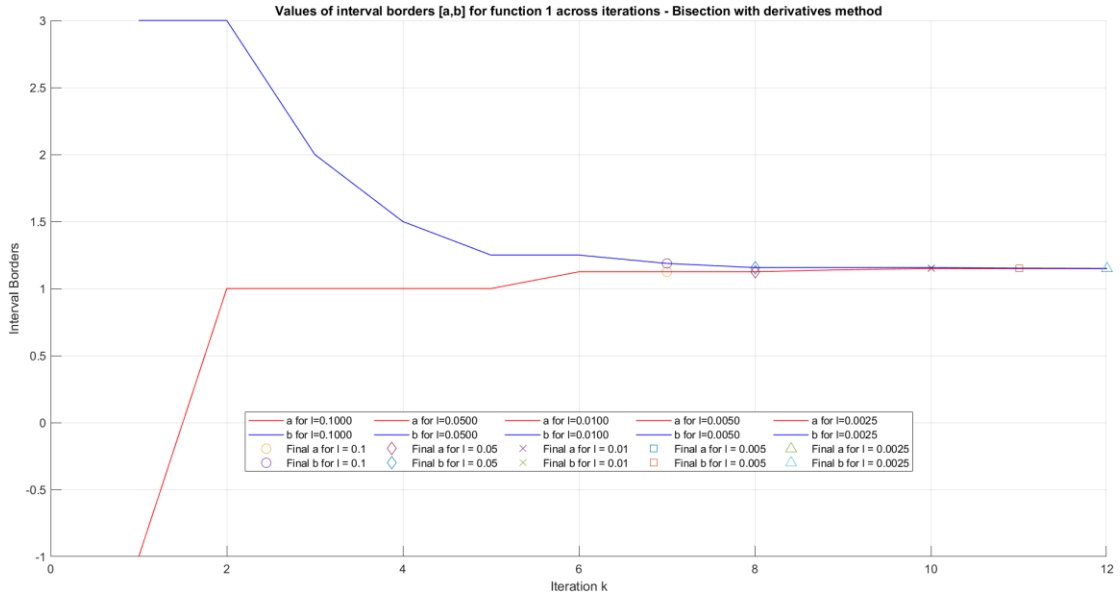

Number of function 1 evaluations for different l values - Bisection with derivatives method
Function 1
Number of function 1 evaluations
l values

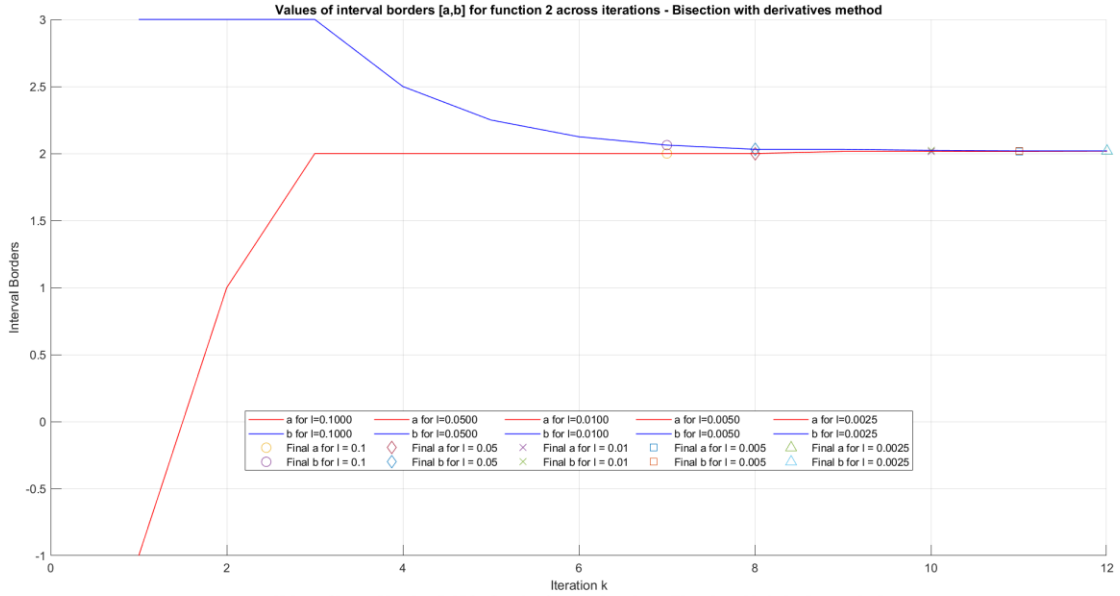

Number of function 2 evaluations for different l values - Bisection with derivatives method
Function 2
Number of function 2 evaluations
l values

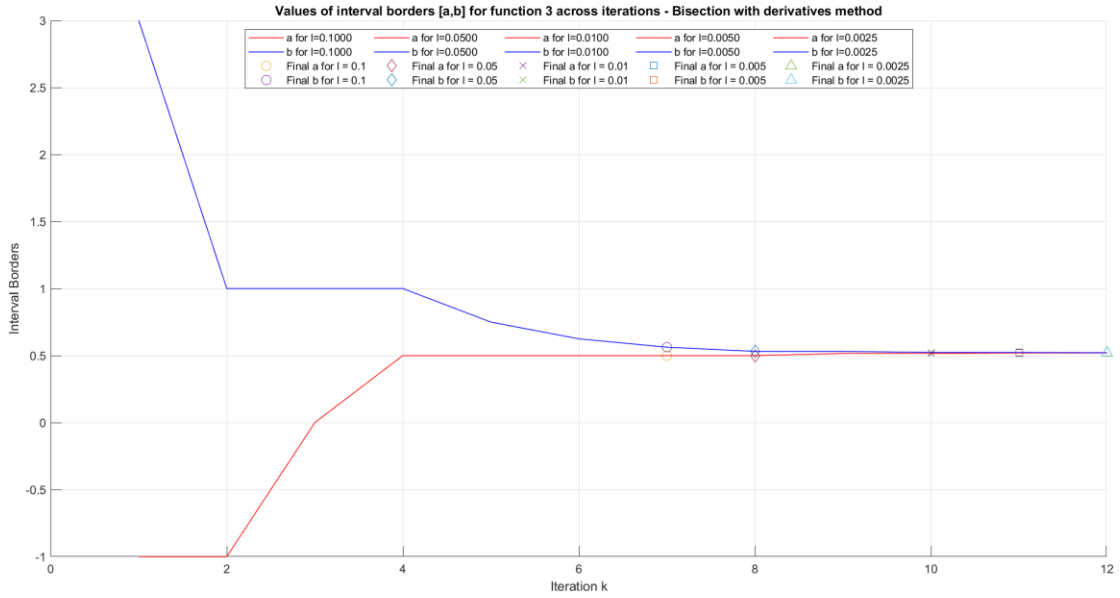

Number of function 3 evaluations for different l values - Bisection with derivatives method
Function 3
Number of function 3 evaluations
l values

Υποπρόβλημα 2: Μεταβολή των άκρων του διαστήματος αναζήτησης σε κάθε επανάληψη, για διάφορες τιμές τελικής ακρίβειας l

Για τις ίδιες τιμές του l που χρησιμοποίησα πριν κάνω τα διαγράμματα για κάθε συνάρτηση των ορίων του διαστήματος αναζήτησης ως προς τον αριθμό επαναλήψεων. Παρατηρώ πως όντως τα διαστήματα συγκλίνουν προς το πραγματικό σημείο ελαχίστου όσο αυξάνονται οι επαναλήψεις.


Values of interval borders [a,b] for function 1 across iterations - Bisection with derivatives method
Interval Borders
Iteration k
a for l=0.1000
a for l=0.0500
a for l=0.0100
a for l=0.0050
a for l=0.0025
b for l=0.1000
b for l=0.0500
b for l=0.0100
b for l=0.0050
b for l=0.0025
Final a for l = 0.1
Final a for l = 0.05
Final a for l = 0.01
Final a for l = 0.005
Final a for l = 0.0025
Final b for l = 0.1
Final b for l = 0.05
Final b for l = 0.01
Final b for l = 0.005
Final b for l = 0.0025


Values of interval borders [a,b] for function 2 across iterations - Bisection with derivatives method
Interval Borders
Iteration k
a for l=0.1000
a for l=0.0500
a for l=0.0100
a for l=0.0050
a for l=0.0025
b for l=0.1000
b for l=0.0500
b for l=0.0100
b for l=0.0050
b for l=0.0025
Final a for l = 0.1
Final a for l = 0.05
Final a for l = 0.01
Final a for l = 0.005
Final a for l = 0.0025
Final b for l = 0.1
Final b for l = 0.05
Final b for l = 0.01
Final b for l = 0.005
Final b for l = 0.0025


Values of interval borders [a,b] for function 3 across iterations - Bisection with derivatives method
Interval Borders
Iteration k
a for l=0.1000
a for l=0.0500
a for l=0.0100
a for l=0.0050
a for l=0.0025
b for l=0.1000
b for l=0.0500
b for l=0.0100
b for l=0.0050
b for l=0.0025
Final a for l = 0.1
Final a for l = 0.05
Final a for l = 0.01
Final a for l = 0.005
Final a for l = 0.0025
Final b for l = 0.1
Final b for l = 0.05
Final b for l = 0.01
Final b for l = 0.005
Final b for l = 0.0025

Σύγκριση των μεθόδων

Ως προς την ταχύτητα σύγκλισης

Παρατηρούμε από τις γραφικές παραστάσεις πως η μέθοδος της διχοτόμου με υπολογισμό παραγώγου συγκλίνει στο τελικό διάστημα πιο γρήγορα από τις υπόλοιπες μεθόδους που δεν χρησιμοποιούνε παραγώγους. Ενδεικτικά αναφέρουμε πως για l=0.0025 η διχοτόμος με παραγώγους τερματίζει σε 12 επαναλήψεις ενώ η διχοτόμος σε 14 ενώ ο χρυσός τομέας και ο Fibonacci σε 17. Ωστόσο για μεγαλύτερες τιμές του l, πχ l=0.1, 0.05, 0.01 η διχοτόμος και η διχοτόμος με παραγώγους έχουν την ίδια ταχύτητα σύγκλισης με 7,8 και 10 επαναλήψεις αντίστοιχα.

Ως προς τον αριθμό υπολογισμών της αντικειμενικής συνάρτησης

Βλέπουμε πως ο αλγόριθμος Fibonacci είναι υπολογιστικά ο πιο αποδοτικός καθώς έχει τον μικρότερο αριθμό επαναλήψεων μεταξύ των 3 μεθόδων που δεν χρησιμοποιούν παραγώγους. Ακολουθεί με μικρή διαφορά ο αλγόριθμος χρυσού τομέα, ενώ η μέθοδος διχοτόμου είναι υπολογιστικά «ασύμφορη» καθώς απαιτεί αρκετούς υπολογισμούς της αντικειμενικής συνάρτησης.

Από την άλλη η μέθοδος διχοτόμου με παραγώγους φαίνεται να είναι μακράν πιο αποδοτική όσον αφορά τον αριθμό αποτιμήσεων της αντικειμενικής συνάρτησης. Παρόλα αυτά η χρήση παραγώγων καθυστερεί αρκετά σε σχέση με τις άλλες μεθόδους την εκτέλεση του προγράμματος κάτι που διαπίστωσα τρέχοντας τον κώδικα. Οπότε σε θέμα χρόνου η μέθοδος αυτή αν και κάνει λιγότερους υπολογισμούς της αντικειμενικής συνάρτησης, χρειάζεται περισσότερο χρόνο για τον κάθε έναν.

Ως προς την ευκολία υλοποίησης

Αυτή η μετρική σχετίζεται με την προσωπική μου εμπειρία στην συγγραφή του κώδικα για κάθε μέθοδο. Η μέθοδος της διχοτόμου ήταν πολύ εύκολη στην υλοποίηση, καθώς και η μέθοδος της διχοτόμου με χρήση παραγώγων. Ελαφρώς πιο δύσκολη ήταν η υλοποίηση του κώδικα για τον αλγόριθμο χρυσού τομέα καθώς προέκυψαν ζητήματα που ανέφερα και παραπάνω σε σχέση με τον έλεγχο των επαναλήψεων. Τέλος η υλοποίηση του αλγορίθμου Fibonacci ήταν αρκετά πιο δύσκολη από τις άλλες καθώς προέκυψαν ζητήματα με το indexing του πίνακα με τους αριθμούς Fibonacci, καθώς και με τον έλεγχο ροής του προγράμματος και τον αριθμό επαναλήψεων. Γενικά για τις μεθόδους Fibonacci και χρυσού τομέα χρειάστηκαν αρκετές ώρες για debugging.