Τεχνικές Βελτιστοποίησης - 1η Εργαστηριακή Άσκηση

Αριστείδης Δασκαλόπουλος (ΑΕΜ: 10640)

6 Νοεμβρίου 2024

Γενική Περιγραφή Προβλήματος

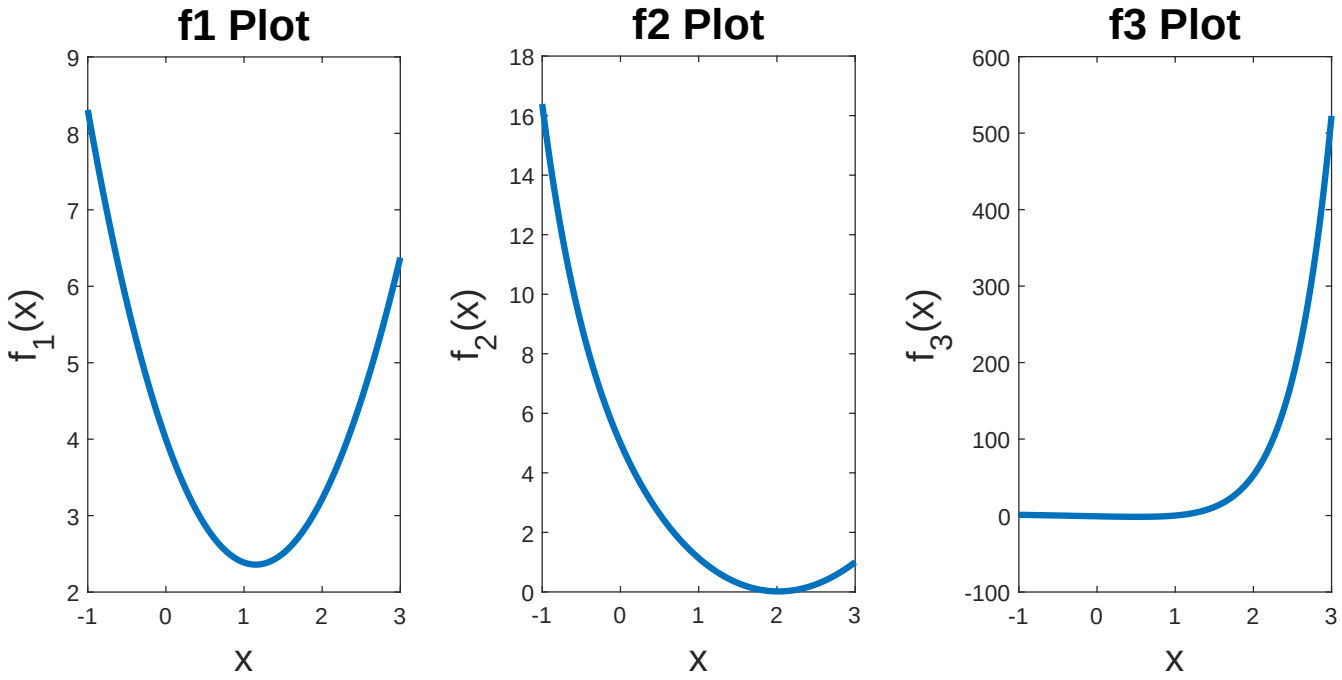
***Στόχος**: Εύρεση ελαχίστου δοσμένης κυρτής συνάρτησης $f(x)$ όταν $x \in [a, b]$.*

Πιο συγκεκριμένα, με αρχικό διάστημα $[-1, 3]$, καλούμαστε να υπολογίσουμε **ένα νέο διάστημα** $[a_k, b_k]$ προδιαγεγραμμένης ακρίβειας $l > 0$, άρα $b_k - a_k \leq l$, στο οποίο *να περιέχεται το ελάχιστο* x^* των συναρτήσεων που ψάχνουμε για τις:

- $f_1(x) = (x-2)^2 + x \cdot ln(x+3)$
- $f_2(x) = e^{-2 \cdot x} + (x-2)^2$
- $f_3(x) = e^x \cdot (x^3 - 1) + (x-1) \cdot sin(x)$

με μεθόδους αναζήτησης ελαχίστου χωρίς την χρήση παραγώγων (*Μέθοδος Διχοτόμου, Μέθοδος Χρυσού Τομέα, Μέθοδος Fibonacci*), αλλά και με μία μέθοδο που κάνει χρήση παραγώγων (*Μέθοδος Διχοτόμου με χρήση παραγώγου*).

Όλες οι παραπάνω μέθοδοι μπορούν να εφαρμοστούν μιας και βασίζονται στο θεώρημα 5.1.1 (του βιβλίου) στο οποίο βασική προϋπόθεση είναι οι συναρτήσεις f_i να είναι **αυστηρά σχεδόν κυρτές** στο διάστημα $[a, b]$ που τις μελετάμε. Οι f_i στο $[-1, 3]$ έχουν γραφικές παραστάσεις:



Σχήμα 1: f_i, για $i = 1, 2, 3$

επομένως η προϋπόθεση που αναφέραμε **ισχύει** (οι συναρτήσεις μας είναι αυστηρά κυρτές, *άρα και αυστηρά σχεδόν κυρτές*, στο $[-1, 3]$).

Τέλος, ο αλγόριθμος με χρήση παραγώγου είναι επίσης εφαρμόσιμος σε όλες τις f_i που έχουμε, καθώς αυτές είναι **παραγωγίσιμες** στο διάστημα που τις μελετάμε (*με γνωστή υπολογίσιμη παράγωγο*).

Στην συνέχεια της αναφοράς για καθεμία από τις μεθόδους θα αναλύσουμε περιεκτικά τον τρόπο λειτουργίας της, θα παρουσιάσουμε/εξηγήσουμε τα αποτελέσματα των ερωτημάτων που ζητούνται ανά θέμα και τέλος θα αναφέρουμε τα πλεονεκτήματα και μειονεκτήματα που εμφανίζει ο κάθε αλγόριθμος - μέθοδος.

Θέμα 1

Παρουσίαση Μεθόδου Διχοτόμου χωρίς χρήση παραγώγων

Η μέθοδος για δεδομένη συνάρτηση $f(x)$ (όταν θα αναφερόμαστε σε συνάρτηση $f(x)$ υποθέτουμε πάντα πως αυτή ικανοποιεί τις προϋποθέσεις που έχουμε προηγούμενος αναφέρει για την αντίστοιχη μέθοδο) ξεκινάει από το αρχικό διάστημα $[a, b]$ - στο οποίο ψάχνουμε να βρούμε το ελάχιστο x^* - και σε κάθε επανάληψη υπολογίζει ένα νέο, μικρότερο του προηγούμενου διάστημα $[a_k, b_k]$ (όπου k η επανάληψη), στο οποίο και πάλι υπάρχει το x^*.

Η μέθοδος σταματάει στην επανάληψη n για την οποία ισχύει το κριτήριο τερματισμού: $b_n – a_n \leq l$, για δεδομένο l (δηλαδή για δεδομένη ακρίβεια τελικού διαστήματος).

Για τον δε υπολογισμό του επόμενου διαστήματος $[a_{k+1}, b_{k+1}]$ σε κάθε επανάληψη:

- Παίρνουμε δυο σημεία $x_{1,k}, x_{2,k} \in [a_k, b_k]$ γύρω από την διχοτόμο του διαστήματος αυτού σε απόσταση ε από αυτήν: $x_{1,k} = \frac{a_k+b_k}{2} - \varepsilon \;\; x_{2,k} = \frac{a_k+b_k}{2} + \varepsilon$
- Υπολογίζουμε τις τιμές της f στα σημεία αυτά $f(x_{1,k})$ $f(x_{2,k})$ και τις συγκρίνουμε:
  - Αν $f(x_{1,k}) < f(x_{2,k}) \Rightarrow a_{k+1} = a_k \;\;,\;\; b_{k+1} = x_{2k}$
  - Αλλιώς $f(x_{1,k}) > f(x_{2,k}) \Rightarrow a_{k+1} = x_{1k} \;\;,\;\; b_{k+1} = b_k$
- Ο αλγόριθμος σταματάει όταν ικανοποιηθεί το κριτήριο τερματισμού

Για να λειτουργήσει σωστά ο αλγόριθμος αυτός πρέπει για το ε να ισχύει: $2 \cdot \varepsilon < l$

Από τα παραπάνω καταλαβαίνουμε πως η αντικειμενική συνάρτηση f πραγματοποιεί σε κάθε επανάληψη 2 υπολογισμούς. Επομένως εφόσον το k εκφράζει την επανάληψη στην οποία είμαστε, θα έχουμε $2 \cdot k$ επαναλήψεις μέχρι και εκείνη την στιγμή (θεωρούμε πως $k = 0$ πριν την πρώτη επανάληψη, δηλαδή ότι $a = a_0$, $b = b_0$).

Στην υλοποίηση της συνάρτησης στο αρχείο `/src/dichotomousMethod.m` το output n εκφράζει τον συνολικό τελικό αριθμό των επαναλήψεων (πχ Αν $n = 3$, συμαίνει ότι έχουν γίνει συνολικά 3 επαναλήψεις στην `while` επομένως $2 \cdot 3 = 6$ υπολογισμοί της αντικειμενική συνάρτηση f. Τότε η συνάρτηση επιστρέφει τόσο το n όσο και όλα τα όρια του διαστήματος a_0, a_1, a_2, a_3 και b_0, b_1, b_2, b_3 με $b_3 - a_3 < l$)

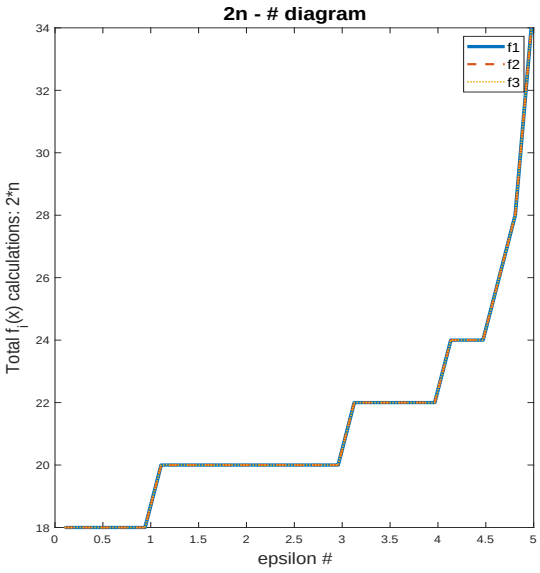
Ζητούμενο 1 - Σταθερό τελικό εύρος αναζήτησης $\mathbf{l = 0.01}$

Παίρνοντας διάφορες τιμές για το ε υπολογίζουμε για το σταθερό l το πλήθος των υπολογισμών της αντικειμενικής συνάρτησης για καθεμιά από τις 3 συναρτήσεις μας, λαμβάνοντας υπόψιν την παραπάνω παρατήρηση σχετικά με την σχέση του πλήθους των επαναλήψεων και του πλήθους των υπολογισμών της αντικειμενικής συνάρτησης.

Για να πάρουμε τιμές για το ε χρησιμοποιούμε τις παρακάτω γραμμές κώδικα (θα μπορούσαμε να είχαμε βάλει και τυχαίες τιμές χειροκίνητα, προσέχοντας πάντα να ικανοποιείται η συνθήκη $2 \cdot \varepsilon < l$ διαφορετικά δεν θα τερματίζει ποτέ ο αλγόριθμος).

```
l = 0.01;
sampleSize = 30;
epsilonList = linspace(l/100, l/2.01, sampleSize);
```

Τα διαγράμματα που παίρνουμε για κάθε συνάρτηση φαίνονται παρακάτω σε κοινό figure με τον οριζόντιο άξονα των ε να είναι σε κλίμακα 10^{-3}:



Παρατηρούμε πως όσο το ε μεγαλώνει και πλησιάζει στην τιμή $l/2$ απαιτούνται περισσότεροι υπολογισμοί της αντικειμενικής συνάρτησης κάτι που κάνει τον αλγόριθμο πιο αγρό (περισσότερο υπολογιστικό κόστος).

Επίσης, παρατηρούμε πως για όλες τις συναρτήσεις βγάζουμε πάντα ακριβώς την ίδια παράσταση, μιας και οι επαναλήψεις του αλγορίθμου εξαρτώνται μόνο από τα l, ε και το αρχικό διάστημα $[a, b]$.

Το γεγονός ότι η γραφική παράσταση εμφανίζει επίπεδα έχει να κάνει με το ότι για πολύ μικρές αλλαγές του ε δεν υπάρχει καμία πρακτική αλλαγή για την δεδομένη ακρίβεια που εκφράζεται από το l.

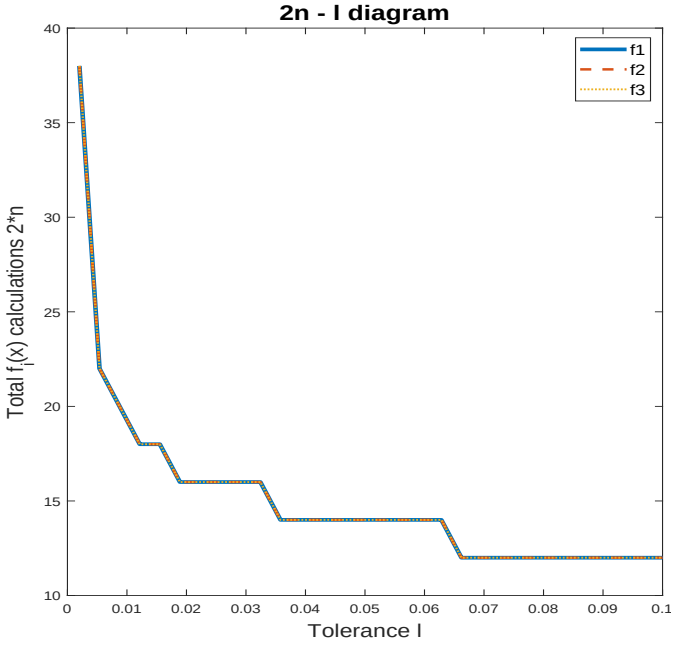
Ζητούμενο 2 - Σταθερό ε = 0.001

Ομοίως, παίρνοντας διάφορες τιμές για το l υπολογίζουμε για σταθερό ε το πλήθος των υπολογισμών της αντικειμενικής συνάρτησης για καθεμιά από τις 3 συναρτήσεις μας.

Για να πάρουμε τιμές για το l χρησιμοποιούμε τις παρακάτω γραμμές κώδικα (προσέχομυμε να ικανοποιείται η συνθήκη $2 \cdot \varepsilon < l$ διαφορετικά δεν θα τερματίζει ποτέ ο αλγόριθμος).

```
epsilon = 0.001;
sampleSize = 30;
lList = linspace(epsilon * 2.01, epsilon * 100, sampleSize);
```

Τα διαγράμματα που παίρνουμε για κάθε συνάρτηση φαίνονται παρακάτω σε κοινό figure:



Παρατηρούμε πως όσο το l μεγαλώνει απαιτούνται συνολικά λιγότεροι υπολογισμοί της αντικειμενικής συνάρτησης διότι με την αύξηση του l ζητάμε μεγαλύτερο τελικό εύρος, άρα και μικρότερη ακρίβεια, συνεπώς απαιτούνται λιγότεροι υπολογισμοί/επαναλήψεις.

Και πάλι, παρατηρούμε πως για όλες τις συναρτήσεις βγάζουμε πάντα ακριβώς την ίδια παράσταση, μιας και οι επαναλήψεις του αλγορίθμου εξαρτώνται μόνο από τα l, ε και το αρχικό διάστημα $[a, b]$.

Το γεγονός ότι η γραφική παράσταση εμφανίζει επίπεδα έχει να κάνει με το ότι για πολύ μικρές αλλαγές του l δεν υπάρχει καμία πρακτική αλλαγή στο εύρος του τελικού διαστήματος (για την δεδομένο ε).

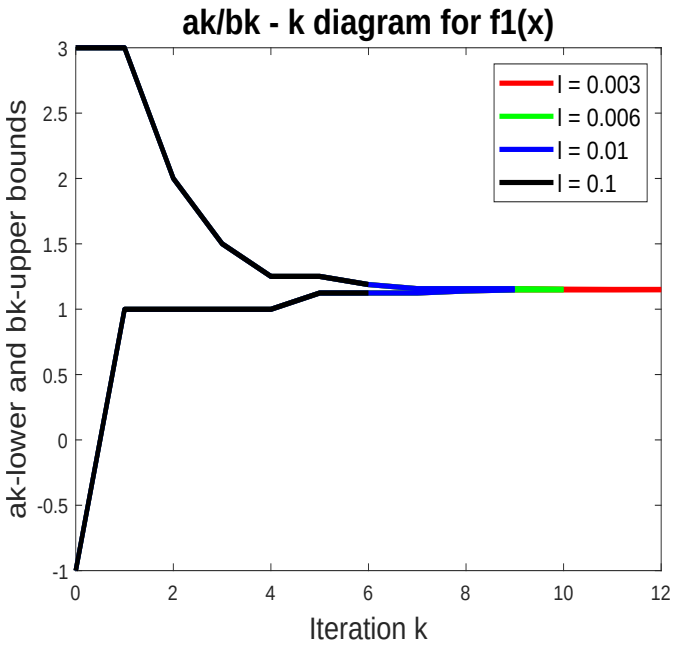
Αν θέλουμε μεγαλύτερη ακρίβεια στις γραφικές παραστάσεις μπορούμε, είτε να πάρουμε περισσότερα σημεία για υπολογισμό, είτε να μειώσουμε το εύρος στον οριζόντιο άξονα.

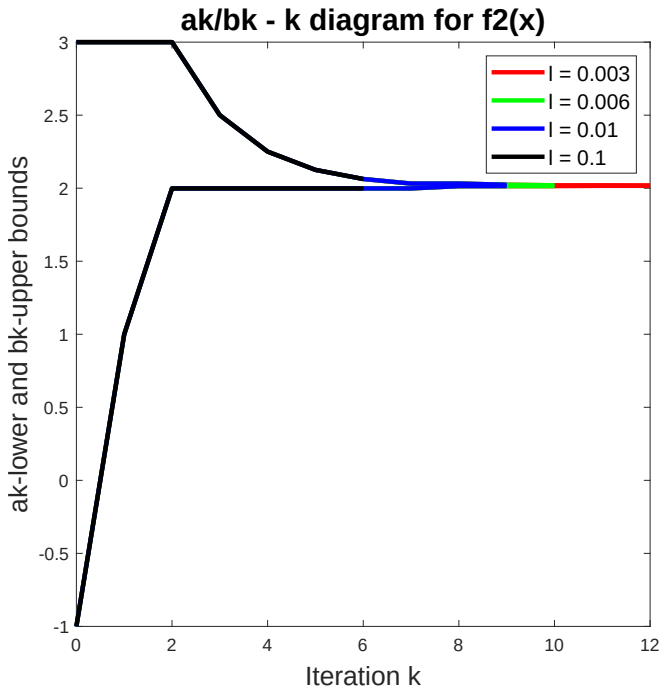
Ζητούμενο 3 - Plot υποδιαστημάτων [ak,bk] για διάφορα l

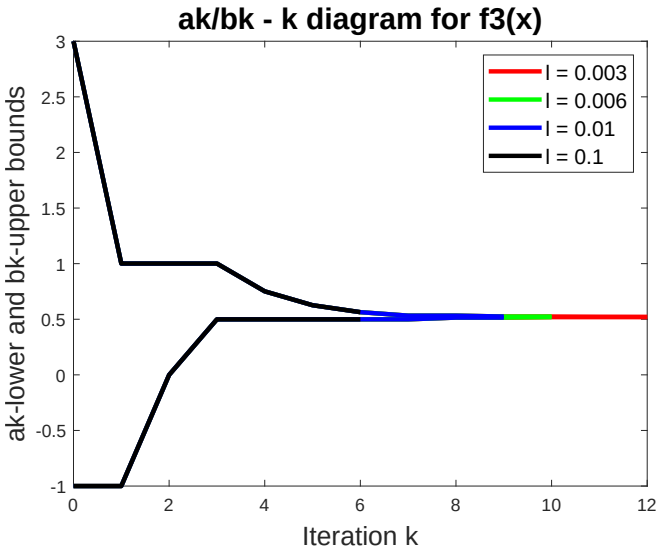
Για διάφορες τιμές του l υπολογίζουμε για σταθερό ε όλα τα άνω και κάτω όρια, a_k και b_k των υποδιαστημάτων. Για κάθε συνάρτηση ξεχωριστά, κάνουμε ένα κοινό plot και για τα δύο όρια για διάφορες τιμές του l.

Για να πάρουμε τιμές για το l χρησιμοποιούμε τις παρακάτω γραμμές κώδικα (προσέχομυμε να ικανοποιείται η συνθήκη $2 \cdot \varepsilon < l$ διαφορετικά δεν θα τερματίζει ποτέ ο αλγόριθμος).

```
epsilon = 0.001;
lValues = [0.003, 0.006, 0.01, 0.1];
```







Τα διαγράμματα που παίρνουμε για κάθε συνάρτηση φαίνονται παραπάνω.

Τα δύο όρια συγκλίνουν - όπως άλλωστε περιμέναμε.

Παρατηρούμε πως όσο το l μειώνεται απαιτούνται συνολικά περισσότεροι υπολογισμοί/επαναλήψεις n, διότι με μικρό l ζητάμε μικρότερο τελικό εύρος, άρα και μεγαλύτερη ακρίβεια, συνεπώς απαιτούνται περισσότεροι υπολογισμοί/επαναλήψεις.

Βλέπουμε πως για κάθε συνάρτηση βγάζουμε πάντα την ίδια παράσταση μόνο που όταν το l αυξάνεται προχωράμε μερικές επαναλήψεις παρακάτω. Αυτό συμβαίνει μιας και το l δεν αλλάζει τα αποτελέσματα των υπολογισμών για δεδομένη $f(x)$ παρά μόνο ελέγχει αν έχουμε φτάσει στην επιθυμητή ακρίβεια για να σταματήσει ο αλγόριθμος.

Είναι εμφανές ότι σε κάθε επανάληψη k είτε πέφτει το άνω όριο, είτε αυξάνεται το κάτω - ποτέ δεν έχουμε αλλαγή ταυτόχρονα και στα δύο.

Σχόλια

Η μέθοδος διχοτόμησης είναι μία απλή στην υλοποίηση και κατανόηση μέθοδος υπολογισμού ελαχίστου. Παρόλα αυτά, σε σχέση με όλες τις υπόλοιπες που θα δούμε παρακάτω είναι υπολογιστικά ασύμφορη - μιας και απαιτεί 2 υπολογισμούς της αντικειμενικής συνάρτησης ανά επανάληψη και κάτι τέτοιο διπλασιάζει την χρονική πολυπλοκότητα του αλγορίθμου.

Θέμα 2

Παρουσίαση Μεθόδου Χρυσού Τομέα

Η μέθοδος για δεδομένη συνάρτηση $f(x)$ ξεκινάει από το αρχικό διάστημα $[a, b]$ - στο οποίο ψάχνουμε να βρούμε το ελάχιστο x^* - και σε κάθε επανάληψη υπολογίζει ένα νέο, μικρότερο του προηγούμενου διάστημα $[a_k, b_k]$ (όπου k η επανάληψη), στο οποίο και πάλι υπάρχει το x^*.

Η μέθοδος σταματάει στην επανάληψη n για την οποία ισχύει το γνωστό κριτήριο τερματισμού: b_n–$a_n \leq l$, για δεδομένο l (δηλαδή για δεδομένη ακρίβεια τελικού διαστήματος).

Για τον δε υπολογισμό του επόμενου διαστήματος $[a_{k+1}, b_{k+1}]$, έχουμε την σχέση $b_{k+1} - a_{k+1} = \gamma(b_k - a_k)$, όπου για να έχουμε "χρύση τομή" προκύπτει η εξίσωση $\gamma^2 + \gamma - 1 = 0 \Rightarrow \gamma = 0.6180...$ και στην πρώτη επανάληψη:

- Παίρνουμε δυο σημεία $x_1 = a + (1-\gamma)(b-a) = b - \gamma(b-a)$, $x_2 = a + \gamma(b-a) \in [a, b]$
- Υπολογίζουμε τις τιμές της f στα σημεία αυτά $f(x_1)$ $f(x_2)$ και μπαίνουμε στον βρόχο while

Σε κάθε επανάληψη της while:

- Συγκρίνουμε $f(x_{1,k})$ και $f(x_{2,k})$:
  - Αν $f(x_{1,k}) < f(x_{2,k}) \Rightarrow a_{k+1} = a_k$, $b_{k+1} = x_{2,k}$ και
    $x_{1,k+1} = a_{k+1} + (1-\gamma)(b_{k+1} - a_{k+1}) = b_{k+1} - \gamma(b_{k+1} - a_{k+1})$, $x_{2,k+1} = x_{1,k}$
  - Αλλιώς $f(x_{1,k}) > f(x_{2,k}) \Rightarrow a_{k+1} = x_{1,k}$, $b_{k+1} = b_k$ και
    $x_{2,k+1} = a_{k+1} + \gamma(b_{k+1} - a_{k+1})$, $x_{1,k+1} = x_{2,k}$
- Ο αλγόριθμος σταματάει όταν ικανοποιηθεί το κριτήριο τερματισμού

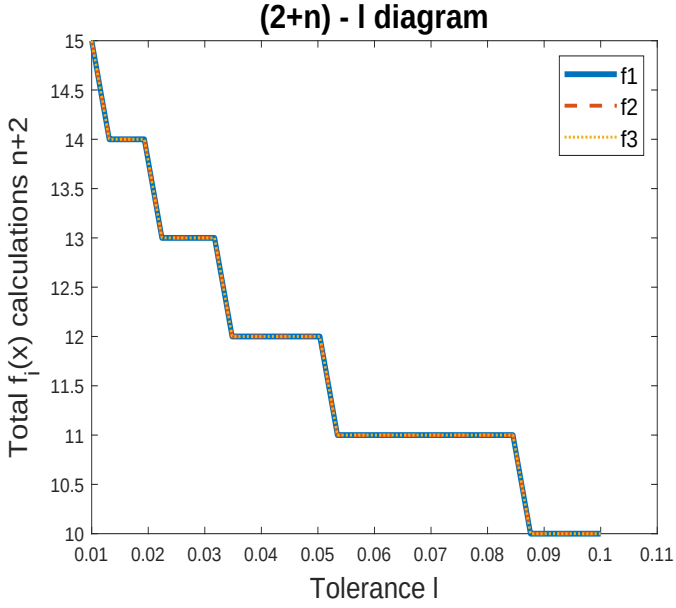
Από τα παραπάνω καταλαβαίνουμε πως η αντικειμενική συνάρτηση f πραγματοποιεί στην πρώτη επανάληψη 2 υπολογισμούς ενώ σε κάθε επόμενη από 1 υπολογισμό μιας και έχουμε σε κάθε επανάληψη $x_{1,k+1} = x_{2,k}$ ή $x_{2,k+1} = x_{1,k}$, άρα γνωστές και ήδη υπολογισμένες τιμές της f σε κάθε περίπτωση. Επομένως, εφόσον το k εκφράζει την επανάληψη από την στιγμή που μπούμε στην while loop, θα έχουμε $2 + k$ υπλογισμούς συνολικά. (Θεωρούμε πως $k = 0$ όταν αρχίζει ο αλγόριθμος, δηλαδή όταν $a = a_0$, $b = b_0$).

Στην υλοποίηση της συνάρτησης το output n εκφράζει τον συνολικό τελικό αριθμό των επαναλήψεων της while loop. (πχ Αν $n = 3$, συμαίνει ότι έχουν γίνει συνολικά 3 επαναλήψεις στην while επομένως $2 + 3 = 5$ υπολογισμοί της αντικειμενική συνάρτηση f. Τότε η συνάρτηση επιστρέφει τόσο το n όσο και όλα τα όρια του διαστήματος a_0, a_1, a_2, a_3 και b_0, b_1, b_2, b_3 με $b_3 - a_3 < l$).

Ο ορισμός που δώσαμε για το n στην συνάρτηση είναι τέτοιος ώστε το $n+1$ να μην εκφράζει τον συνολικό αριθμό των υπολογισμών της αντικειμενικής.

Ζητούμενο 1 - Μεταβολή l

Για διάφορες τιμές για το l υπολογίζουμε το πλήθος των υπολογισμών της αντικειμενικής συνάρτησης για καθεμιά από τις 3 συναρτήσεις μας, λαμβάνοντας υπόψιν την παραπάνω παρατήρηση σχετικά με την σχέση του πλήθους των επαναλήψεων της while loop και του πλήθους των υπολογισμών της αντικειμενικής συνάρτησης.
Το διάγραμμα που παίρνουμε για κάθε συνάρτηση φαίνονται παρακάτω σε κοινό figure:



Παρατηρούμε πως όσο το l μεγαλώνει απαιτούνται συνολικά λιγότεροι υπολογισμοί της αντικειμενικής συνάρτησης διότι με την αύξηση του l ζητάμε μεγαλύτερο τελικό εύρος, άρα και μικρότερη ακρίβεια, συνεπώς απαιτούνται λιγότεροι υπολογισμοί/επαναλήψεις.

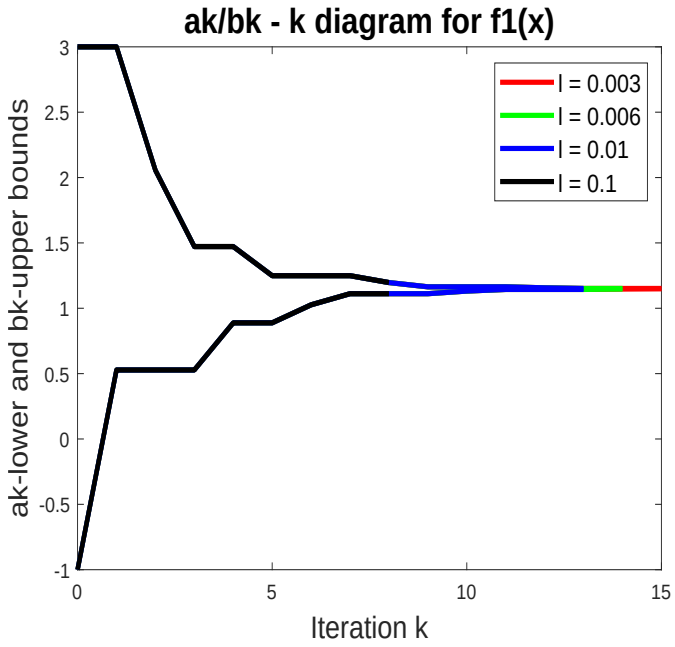
Για όλες τις συναρτήσεις βγάζουμε πάντα ακριβώς την ίδια παράσταση, μιας και οι επαναλήψεις του αλγορίθμου εξαρτώνται μόνο από το l και το αρχικό διάστημα $[a, b]$.

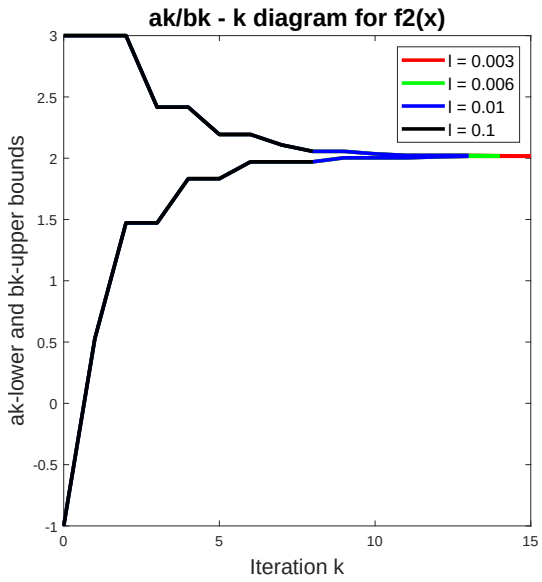
Το γεγονός ότι η γραφική παράσταση εμφανίζει επίπεδα έχει να κάνει με το ότι για πολύ μικρές αλλαγές του l δεν υπάρχει καμία πρακτική αλλαγή στο εύρος του τελικού διαστήματος.

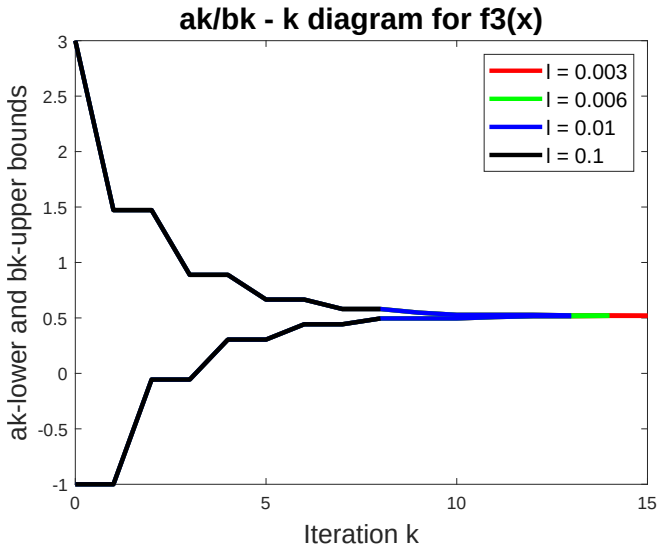
Αν θέλουμε μεγαλύτερη ακρίβεια στις γραφικές παραστάσεις μπορούμε, είτε να πάρουμε περισσότερα σημεία για υπολογισμό, είτε να μειώσουμε το εύρος στον οριζόντιο άξονα.

Ζητούμενο 2 - Plot υποδιαστημάτων [ak,bk] για διάφορα l

Για διάφορες τιμές του l υπολογίζουμε όλα τα άνω και κάτω όρια, a_k και b_k των υποδιαστημάτων. Για κάθε συνάρτηση ξεχωριστά, κάνουμε ένα κοινό plot και για τα δύο όρια για διάφορες τιμές του l.
Τα διαγράμματα που παίρνουμε για κάθε συνάρτηση φαίνονται παρακάτω:







Τα δύο όρια συγκλίνουν - όπως άλλωστε περιμέναμε.

Παρατηρούμε πως όσο το l μειώνεται απαιτούνται συνολικά περισσότεροι υπολογισμοί/επαναλήψεις n, διότι με μικρό l ζητάμε μικρότερο τελικό εύρος, άρα και μεγαλύτερη ακρίβεια, συνεπώς απαιτούνται περισσότεροι υπολογισμοί/επαναλήψεις.

Βλέπουμε πως για κάθε συνάρτηση βγάζουμε πάντα την ίδια παράσταση μόνο που όταν το l αυξάνεται προχωράμε μερικές επαναλήψεις παρακάτω. Αυτό συμβαίνει μιας και το l δεν αλλάζει τα αποτελέσματα των υπολογισμών για δεδομένη $f(x)$ παρά μόνο ελέγχει αν έχουμε φτάσει στην επιθυμητή ακρίβεια για να σταματήσει ο αλγόριθμος.

Είναι εμφανές ότι σε κάθε επανάληψη k είτε πέφτει το άνω όριο, είτε αυξάνεται το κάτω - ποτέ δεν έχουμε αλλαγή ταυτόχρονα και στα δύο.

Σχόλια

Η μέθοδος του χρυσού τομέα χάρη στον τρόπο με τον οποίο αλλάζει ανά επανάληψη το διάστημα $[a_k, b_k]$ - όπως αναφέραμε αναλυτικότερα και στην παρουσίαση της μεθόδου - δίνει την δυνατότητα ενός μόνο υπολογισμού της αντικειμενικής συνάρτησης μετά τους πρώτους 2 υπολογισμούς που απαιτούνται κατά την εκκίνηση του αλγορίθμου. Εμφανίζονται βέβαια και πάλι αρκετές επαναλήψεις για να δώσει ένα ικανοποιητικό αποτέλεσμα.

Θέμα 3

Παρουσίαση Μεθόδου Fibonacci

Η μέθοδος και πάλι για δεδομένη συνάρτηση $f(x)$ ξεκινάει από το αρχικό διάστημα $[a, b]$ - στο οποίο ψάχνουμε να βρούμε το ελάχιστο x^* - και σε κάθε επανάληψη υπολογίζει ένα νέο, μικρότερο του προηγούμενου διάστημα $[a_k, b_k]$ (όπου k η επανάληψη), στο οποίο και πάλι υπάρχει το x^*.

Η μέθοδος σταματάει στην επανάληψη n για την οποία ισχύει: $(b–a)/l < F_n$, για δεδομένο l και F_n ο n-στος αριθμός Fibonacci. (Θεωρούμε ακολουθία Fibonacci με $F_0 = F_1 = 1$).

Χρησιμοποιούμε αυτό το κριτήριο, που είναι ισοδύναμο με αυτό που ως τώρα έχουμε παρουσιάσει, μιας και θέλουμε εκ των προτέρων να γνωρίζουμε όλους τους αριθμούς Fibonacci που θα χρειαστούμε μίας και αυτός ο τρόπος είναι υπολογιστικά βέλτιστος (καθός κλήση της συνάρτησης fibonacci είναι "ακριβή").

Για τον δε υπολογισμό του επόμενου διαστήματος $[a_{k+1}, b_{k+1}]$ στην πρώτη επανάληψη:

- Παίρνουμε δυο σημεία $x_1 = a + \frac{F_{n-2}}{F_n}(b-a)$, $x_2 = a + \frac{F_{n-1}}{F_n}(b-a)$
- Υπολογίζουμε τις τιμές της f στα σημεία αυτά $f(x_1)$ $f(x_2)$ και μπαίνουμε στον βρόχο for

Σε κάθε επανάληψη της for:

- Συγκρίνουμε $f(x_{1,k})$ και $f(x_{2,k})$:
  - Αν $f(x_{1,k}) < f(x_{2,k}) \Rightarrow a_{k+1} = a_k$, $b_{k+1} = x_{2,k}$ και $x_{1,k+1} = a_{k+1} + \frac{F_{n-k-1}}{F_{n-k+1}}(b_{k+1} - a_{k+1})$, $x_{2,k+1} = x_{1,k}$
  - Αλλιώς $f(x_{1,k}) > f(x_{2,k}) \Rightarrow a_{k+1} = x_{1,k}$, $b_{k+1} = b_k$ και $x_{2,k+1} = a_{k+1} + \frac{F_{n-k}}{F_{n-k+1}}(b_{k+1} - a_{k+1})$, $x_{1,k+1} = x_{2,k}$
- Ο αλγόριθμος σταματάει όταν τελειώσει η for loop. (Τότε θα έχει ικανοποιηθεί το κριτήριο τερματισμού)

Από τα παραπάνω καταλαβαίνουμε πως η αντικειμενική συνάρτηση f χρειάζεται στην πρώτη επανάληψη 2 υπολογισμούς ενώ σε κάθε επόμενη από 1 υπολογισμό μιας και έχουμε σε κάθε επανάληψη $x_{1,k+1} = x_{2,k}$ ή $x_{2,k+1} = x_{1,k}$, άρα γνωστές και ήδη υπολογισμένες τιμές της f σε κάθε περίπτωση. Επομένως, εφόσον το k εκφράζει την επανάληψη από την στιγμή που μπούμε στην for loop, θα έχουμε $2 + k$ υπλογισμούς συνολικά. (Θεωρούμε πως $k = 0$ όταν αρχίζει ο αλγόριθμος, δηλαδή όταν $a = a_0$, $b = b_0$).

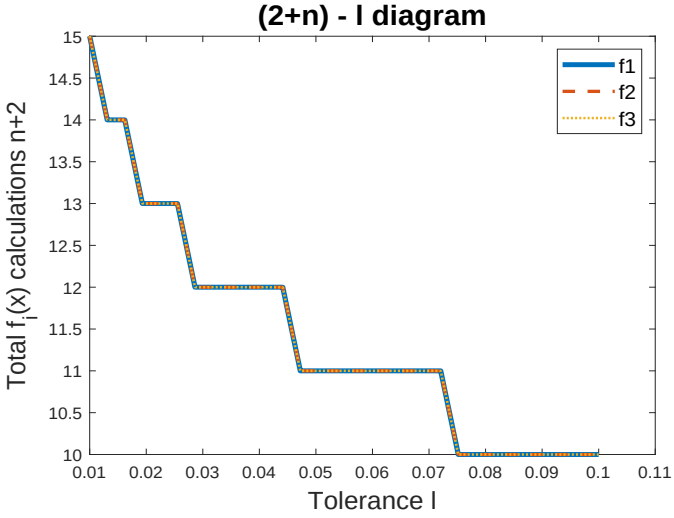
Στην υλοποίηση της συνάρτησης το output n εκφράζει τον συνολικό τελικό αριθμό των επαναλήψεων της for loop. (πχ Αν $n = 3$, συμαίνει ότι έχουν γίνει συνολικά 3 επαναλήψεις στην for επομένως $2 + 3 = 5$ υπολογισμοί της αντικειμενική συνάρτηση f. Τότε η συνάρτηση επιστρέφει τόσο το n όσο και όλα τα όρια του διαστήματος a_0, a_1, a_2, a_3 και b_0, b_1, b_2, b_3 με $b_3 - a_3 < l$).

Ο ορισμός που δώσαμε για το n στο output της συνάρτησης δεν είναι το θεωρητικό n για το οποίο ισχύει $(b–a)/l < F_n$, αλλά ο αριθμός των επαναλήψεων της for loop.

Ζητούμενο 1 - Μεταβολή l

Για διάφορες τιμές για το l υπολογίζουμε το πλήθος των υπολογισμών της αντικειμενικής συνάρτησης για καθεμιά από τις 3 συναρτήσεις μας, λαμβάνοντας υπόψιν την παραπάνω παρατήρηση σχετικά με την σχέση του πλήθους των επαναλήψεων της for loop και του πλήθους των υπολογισμών της αντικειμενικής συνάρτησης.

Το διάγραμμα που παίρνουμε για κάθε συνάρτηση φαίνονται παρακάτω σε κοινό figure:



Παρατηρούμε πως όσο το l μεγαλώνει απαιτούνται συνολικά λιγότεροι υπολογισμοί της αντικειμενικής συνάρτησης διότι με την αύξηση του l ζητάμε μεγαλύτερο τελικό εύρος, άρα και μικρότερη ακρίβεια, συνεπώς απαιτούνται λιγότεροι υπολογισμοί/επαναλήψεις.

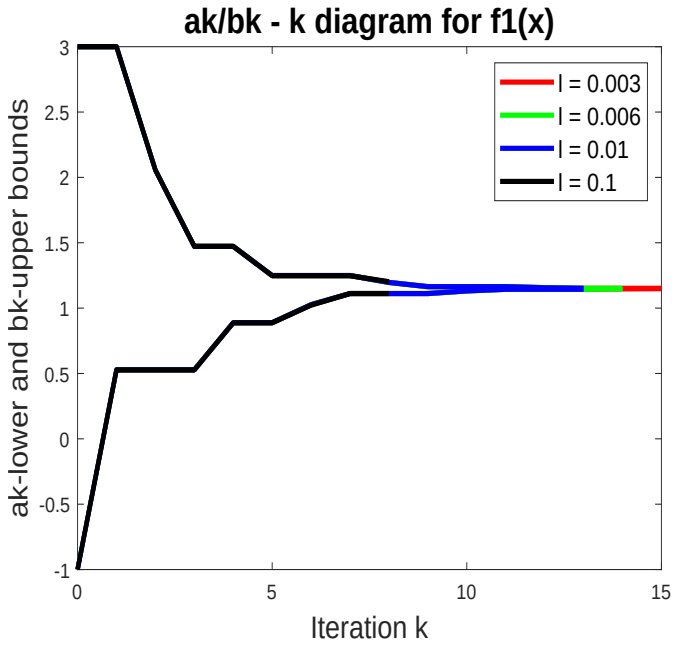
Για όλες τις συναρτήσεις βγάζουμε πάντα ακριβώς την ίδια παράσταση, μιας και οι επαναλήψεις του αλγορίθμου εξαρτώνται μόνο από το l και το αρχικό διάστημα $[a, b]$.

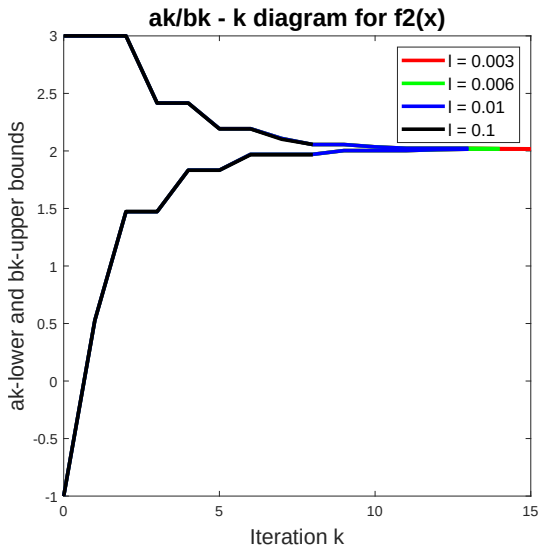
Το γεγονός ότι η γραφική παράσταση εμφανίζει επίπεδα έχει να κάνει με το ότι για πολύ μικρές αλλαγές του l δεν υπάρχει καμία πρακτική αλλαγή στο εύρος του τελικού διαστήματος.

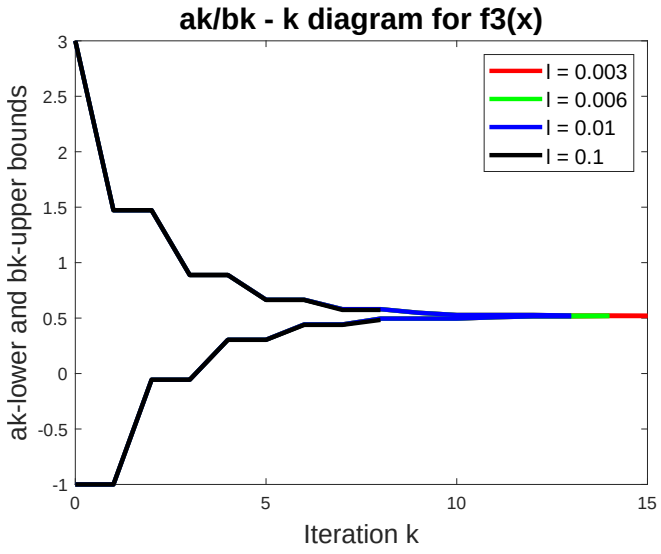
Αν θέλουμε μεγαλύτερη ακρίβεια στις γραφικές παραστάσεις μπορούμε, είτε να πάρουμε περισσότερα σημεία για υπολογισμό, είτε να μειώσουμε το εύρος στον οριζόντιο άξονα.

Ζητούμενο 2 - Plot υποδιαστημάτων [ak,bk] για διάφορα l

Για διάφορες τιμές του l υπολογίζουμε όλα τα άνω και κάτω όρια, a_k και b_k των υποδιαστημάτων. Για κάθε συνάρτηση ξεχωριστά, κάνουμε ένα κοινό plot και για τα δύο όρια για διάφορες τιμές του l.
Τα διαγράμματα που παίρνουμε για κάθε συνάρτηση φαίνονται παρακάτω:







Τα δύο όρια συγκλίνουν - όπως άλλωστε περιμέναμε.

Παρατηρούμε πως όσο το l μειώνεται απαιτούνται συνολικά περισσότεροι υπολογισμοί/επαναλήψεις n, διότι με μικρό l ζητάμε μικρότερο τελικό εύρος, άρα και μεγαλύτερη ακρίβεια, συνεπώς απαιτούνται περισσότεροι υπολογισμοί/επαναλήψεις.

Βλέπουμε πως για κάθε συνάρτηση βγάζουμε πάντα την ίδια παράσταση μόνο που όταν το l αυξάνεται προχωράμε μερικές επαναλήψεις παρακάτω. Αυτό συμβαίνει μιας και το l δεν αλλάζει τα αποτελέσματα των υπολογισμών για δεδομένη $f(x)$ παρά μόνο ελέγχει αν έχουμε φτάσει στην επιθυμητή ακρίβεια για να σταματήσει ο αλγόριθμος.

Είναι εμφανές ότι σε κάθε επανάληψη k είτε πέφτει το άνω όριο, είτε αυξάνεται το κάτω - ποτέ δεν έχουμε αλλαγή ταυτόχρονα και στα δύο.

Σχόλια

Η μέθοδος Fibonacci έχει το πλεονέκτημα ενός μόνο υπολογισμού της αντικειμενικής συνάρτησης μετά τους πρώτους 2 υπολογισμούς που απαιτούνται κατά την εκκίνηση του αλγορίθμου, όπως ακριβώς και η μέθοδος του χρυσού τομέα. Γενικά απαιτεί λιγότερες επαναλήψεις από την μέθοδο του χρυσού τομέα (αλλά το αποτέλεσμα στην πράξη δεν βελτιώνει αισθητά την χρονική πολυπλοκότητα), ενώ ταυτόχρονα δίνει την δυνατότητα να γνωρίζουμε εκ των προτέρων πόσες επαναλήψεις n θα χρειαστούν για δεδομένη ακρίβεια. Το βασικό ελάττωμα είναι η εξαιρετικά μεγάλη χωρική πολυπλοκότητα του αλγορίθμου μιας και πρέπει να βρεθούν και να αποθηκευτούν όλοι οι αριθμοί Fibonacci μέχρι και τον n. Το να μην αποθηκεύσουμε καθόλου τους αριθμούς Fibonacci και να τους υπολογίζουμε κάθε φορά αποτελεί κακή προγραμματιστική τεχνική λόγο της αναδρομικής σχέσεις που συνδέει τους αριθμούς Fibonacci.

Θέμα 4

Παρουσίαση Μεθόδου Διχοτόμου με χρήση παραγώγου

Η μέθοδος για δεδομένη συνάρτηση $f(x)$ ξεκινάει από το αρχικό διάστημα $[a, b]$ - στο οποίο ψάχνουμε να βρούμε το ελάχιστο x^* - και σε κάθε επανάληψη υπολογίζει ένα νέο, μικρότερο του προηγούμενου διάστημα $[a_k, b_k]$ (όπου k η επανάληψη), στο οποίο και πάλι υπάρχει το x^*. Προφανώς για να μπορεί να εφαρμοστεί αυτή η μέθοδος στις συναρτήσεις, πρέπει αυτές να είναι παραγωγίσιμες στο διάστημα που τις μελετάμε (με γνωστή υπολογίσιμη παράγωγο).

Η μέθοδος σταματάει στην επανάληψη n για την οποία ισχύει το γνωστό κριτήριο τερματισμού: $b_n - a_n \leq l$, για δεδομένο l (δηλαδή για δεδομένη ακρίβεια τελικού διαστήματος). Το τελικό n των επαναλήψεων πρέπει να ικανοποιεί την σχέση $(\frac{1}{2})^n \leq \frac{l}{b-a}$

Έχοντας ήδη υπολογίσει την παράγωγο της $f(x)$ για να βρούμε το επόμενο διάστημα $[a_{k+1}, b_{k+1}]$ σε κάθε επανάληψη της while:

- Υπολογίζουμε το $x_k = (a_k + b_k)/2$ που είναι το μέσο του διαστήματος $[a_k, b_k]$ και έπειτα το $\frac{df(x)}{dx}|_{x=x_k}$
  - Αν $\frac{df(x)}{dx}|_{x=x_k} < 0 \Rightarrow a_{k+1} = x_k \ , \ b_{k+1} = b_k$
  - Αν $\frac{df(x)}{dx}|_{x=x_k} > 0 \Rightarrow a_{k+1} = a_k \ , \ b_{k+1} = x_k$
  - Αν $\frac{df(x)}{dx}|_{x=x_k} = 0 \Rightarrow a_{k+1} = x_k \ , \ b_{k+1} = x_k$ και τερματίζει την while loop μιας και βρήκαμε το ελάχιστο (στα ακρότατα έχουμε μηδενική παράγωγο). Στον υπολογιστή δεν γίνεται ποτέ να έχω στην πράξη ισότητα με μηδέν όταν δεν χρησιμοποιώ integer. Μπορώ επομένως να δηλώσω ότι αν $|\frac{df(x)}{dx}|_{x=x_k}| < 10^{-20}$ τότε ισχύει πρακτικά ισότητα με μηδέν. Αυτόν τον έλεγχο πρέπει να τον κάνω πρώτα.
- Ο αλγόριθμος σταματάει όταν ικανοποιηθεί το κριτήριο τερματισμού

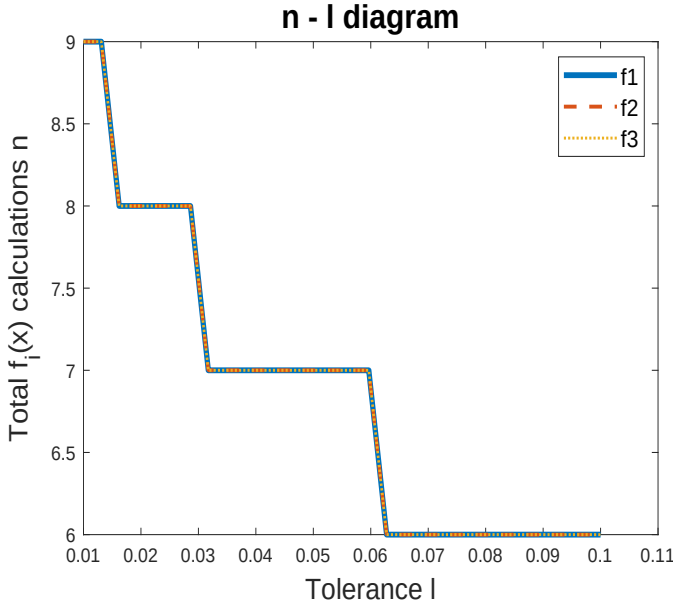
Από τα παραπάνω καταλαβαίνουμε πως η αντικειμενική συνάρτηση (για την αρκίβεια η παράγωγός της) πραγματοποιεί σε κάθε επανάληψη από έναν υπολογισμό. Επομένως, εφόσον το k εκφράζει την επανάληψη από την στιγμή που μπούμε στην while loop, θα έχουμε k υπλογισμούς συνολικά. (Θεωρούμε πως $k = 0$ όταν αρχίζει ο αλγόριθμος, δηλαδή όταν $a = a_0$, $b = b_0$).

Στην υλοποίηση της συνάρτησης το output n εκφράζει τον συνολικό τελικό αριθμό των επαναλήψεων της while loop. (πχ Αν $n = 3$, συμαίνει ότι έχουν γίνει συνολικά 3 επαναλήψεις στην while επομένως 3 υπολογισμοί της παραγώγου της συνάρτησης f. Η συνάρτηση της υλοποίησης επιστρέφει τόσο το n όσο και όλα τα όρια του διαστήματος a_0, a_1, a_2, a_3 και b_0, b_1, b_2, b_3 με $b_3 - a_3 < l$).

Ζητούμενο 1 - Μεταβολή l

Για διάφορες τιμές για το l υπολογίζουμε το πλήθος των υπολογισμών της παραγώγου της συνάρτησης για καθεμιά από τις 3 συναρτήσεις μας.

Το διάγραμμα που παίρνουμε για κάθε συνάρτηση φαίνονται παρακάτω σε κοινό figure:



Παρατηρούμε πως όσο το l μεγαλώνει απαιτούνται συνολικά λιγότεροι υπολογισμοί της αντικειμενικής συνάρτησης

διότι με την αύξηση του l ζητάμε μεγαλύτερο τελικό εύρος, άρα και μικρότερη ακρίβεια, συνεπώς απαιτούνται λιγότεροι υπολογισμοί/επαναλήψεις.

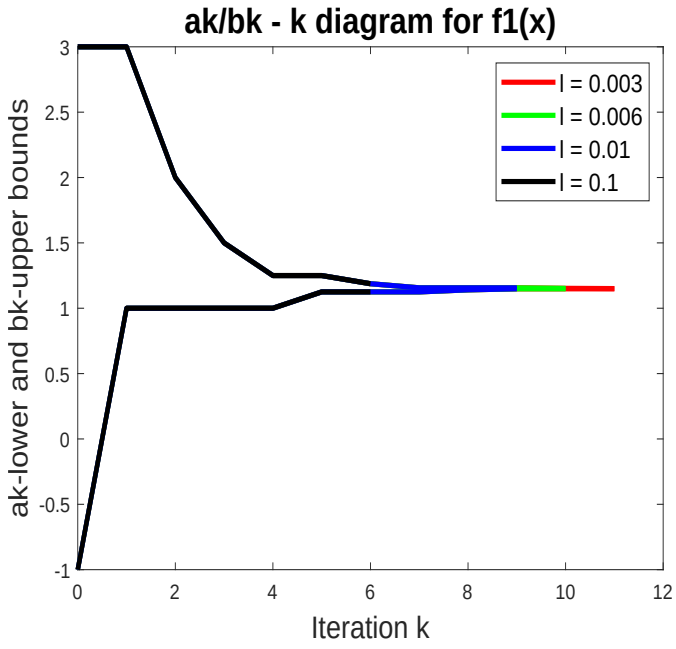
Για όλες τις συναρτήσεις βγάζουμε πάντα ακριβώς την ίδια παράσταση, μιας και οι επαναλήψεις του αλγορίθμου εξαρτώνται μόνο από το l και το αρχικό διάστημα $[a, b]$.

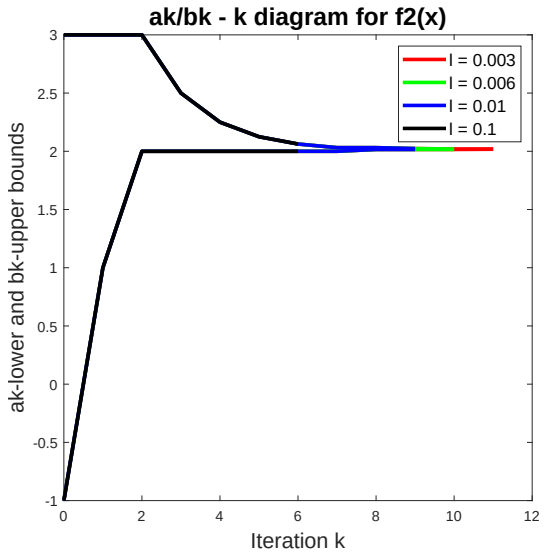
Το γεγονός ότι η γραφική παράσταση εμφανίζει επίπεδα έχει να κάνει με το ότι για πολύ μικρές αλλαγές του l δεν υπάρχει καμία πρακτική αλλαγή στο εύρος του τελικού διαστήματος.

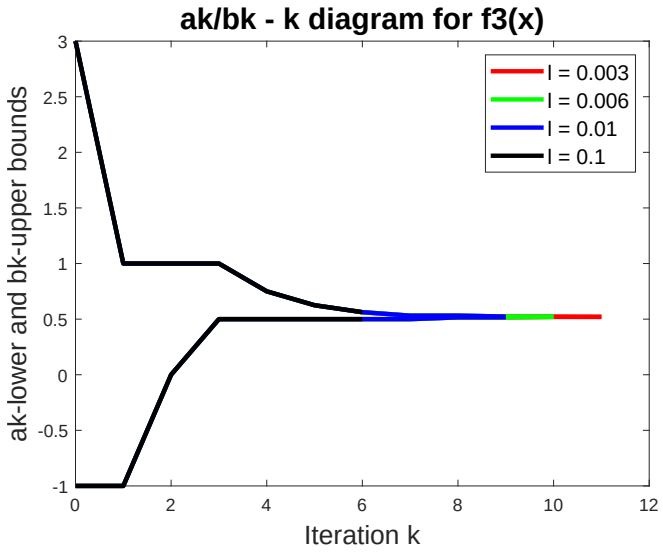
Αν θέλουμε μεγαλύτερη ακρίβεια στις γραφικές παραστάσεις μπορούμε, είτε να πάρουμε περισσότερα σημεία για υπολογισμό, είτε να μειώσουμε το εύρος στον οριζόντιο άξονα.

Ζητούμενο 2 - Plot υποδιαστημάτων [ak,bk] για διάφορα l

Για διάφορες τιμές του l υπολογίζουμε όλα τα άνω και κάτω όρια, a_k και b_k των υποδιαστημάτων. Για κάθε συνάρτηση ξεχωριστά, κάνουμε ένα κοινό plot και για τα δύο όρια για διάφορες τιμές του l.
Τα διαγράμματα που παίρνουμε για κάθε συνάρτηση φαίνονται παρακάτω:







Τα δύο όρια συγκλίνουν - όπως άλλωστε περιμέναμε.

Παρατηρούμε πως όσο το l μειώνεται απαιτούνται συνολικά περισσότεροι υπολογισμοί/επαναλήψεις n, διότι με μικρό l ζητάμε μικρότερο τελικό εύρος, άρα και μεγαλύτερη ακρίβεια, συνεπώς απαιτούνται περισσότεροι υπολογισμοί/επαναλήψεις.

Βλέπουμε πως για κάθε συνάρτηση βγάζουμε πάντα την ίδια παράσταση μόνο που όταν το l αυξάνεται προχωράμε μερικές επαναλήψεις παρακάτω. Αυτό συμβαίνει μιας και το l δεν αλλάζει τα αποτελέσματα των υπολογισμών για δεδομένη $f(x)$ παρά μόνο ελέγχει αν έχουμε φτάσει στην επιθυμητή ακρίβεια για να σταματήσει ο αλγόριθμος.

Είναι εμφανές ότι σε κάθε επανάληψη k είτε πέφτει το άνω όριο, είτε αυξάνεται το κάτω - ποτέ δεν έχουμε αλλαγή ταυτόχρονα και στα δύο.

Σχόλια

Η μέθοδος της διχοτόμου με χρήση παραγώγου είναι η ταχύτερη μέθοδος που εξετάζεται σε αυτήν την εργασία για την εύρεση ελαχίστου. Η πολυπλοκότητά της είναι χαμηλή τόσο χωρικά όσο και χρονικά ενώ και σε αυτή την περίπτωση - όπως και στην μέθοδο Fibonacci - έχουμε την δυνατότητα να γνωρίζουμε εκ των προτέρων πόσες επαναλήψεις θα απαιτηθούν για δεδομένη ακρίβεια (κάτι τέτοιο είναι χρήσιμο σε περιπτώσεις που απαιτούμε μεγάλη ακρίβεια και θέλουμε προτού τρέξουμε τον αλγόριθμο να γνωρίζουμε πόσα βήματα/χρόνος θα χρειαστούν). Το μεγάλο μειονέκτημα της μεθόδου αυτής είναι ότι απαιτεί η συνάρτηση να είναι παραγωγίσιμη στο διάστημα που την μελετάμε (και να μπορούμε φυσικά να υπολογίσουμε την παράγωγό της σε κάθε σημείο του διαστήματος αυτού). Δεν έχουμε όμως πάντα τέτοιες συναρτήσεις οπότε η μέθοδος αυτή έχει πιο περιορισμένη χρήση.

Γενικά Σχόλια

Είναι φανερό από όλη την ανάλυση ότι όλες οι μέθοδοι μπορούν να συγκλίνουν αποτελεσματικά στις πραγματικές λύσεις (οι πραγματικές λύσεις υπολογίζονται στο τέλος κάθε κώδικα στο 'extra' section), με ταχύτερο αλγόριθμο αυτόν που κάνει χρήση της παραγώγου.

Αν η εύρεση της παραγώγου δεν αποτελεί επιλογή (η παράγωγος είναι περίπλοκη και ο υπολογισμός της κοστοβόρος ή η συνάρτηση δεν είναι παραγωγίσιμη σε ολόκληρο το διάστημα) τότε οι αμέσως καλύτερες επιλογές είναι οι μέθοδος Fibonacci και μέθοδος του χρυσού τομέα, επιλέγοντας ανάλογα με το αν έχουμε περιορισμούς στην χωρική πολυπλοκότητα και αν οι τιμές των επαναλήψεων (άρα και των αριθμών Fibonacci που θα χρειαστούν) που θα απαιτηθούν είναι υψηλές. Αν οι αριθμοί Fibonacci που απαιτούνται είναι μεγάλοι θα αρχίσουμε να έχουμε αισθητά μεγάλη καθυστέρηση για τον υπολογισμό τους οπότε θα πρέπει να αποφύγουμε την μέθοδο αυτή και να πάμε με του χρυσού τομέα.