Τεχνικές Βελτιστοποίησης

Ελαχιστοποίηση κυρτής συνάρτησης μιας μεταβλητής σε δοσμένο διάστημα

Θεόδωρος Κατζάλης
ΑΕΜ: 9282
katzalis@auth.gr

17/11/2021

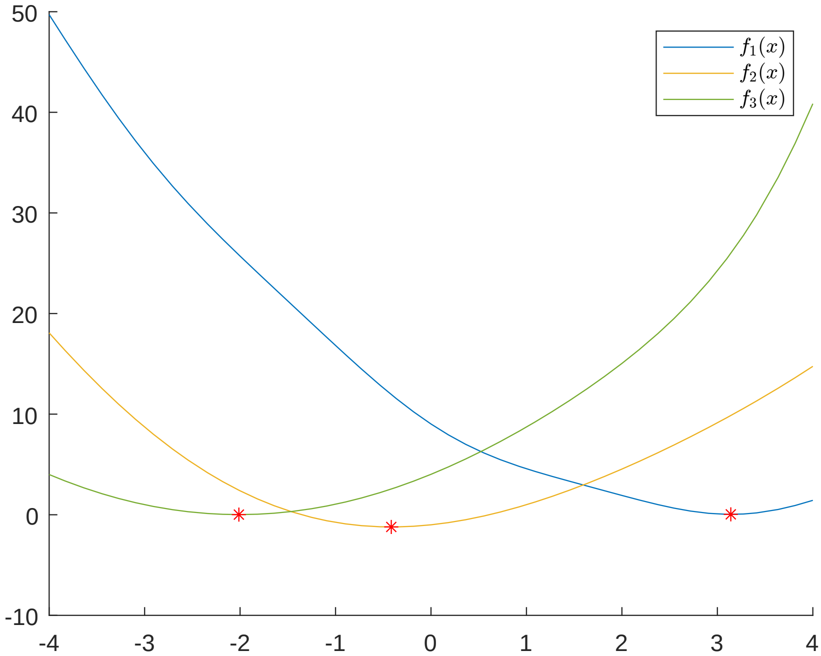
Περιεχόμενα

Εισαγωγή

Σκοπός αυτής της εργασίας είναι η μελέτη διάφορων μεθόδων ελαχιστοποίησης κυρτής συνάρτησης $f(x)$ σε δοσμένο διάστημα $[\alpha,\beta]$. Οι συναρτήσεις που θα μελετηθούν με αρχικό διάστημα $[-4,4]$ είναι οι εξής:

$$f_1(x) = (x-3)^2 + sin^2(x+3) \tag{1}$$
$$f_2(x) = (x-1)cos(\frac{1}{2}x) + x^2 \tag{2}$$
$$f_3(x) = (x+2)^2 + e^{x-2}sin(x+3) \tag{3}$$

Στο παρακάτω γράφημα, με κόκκινο δείκτη απεικονίζεται το ελάχιστο σημείο x^* το οποίο επιθυμούμε να προσεγγίσουμε για την κάθε συνάρτηση.



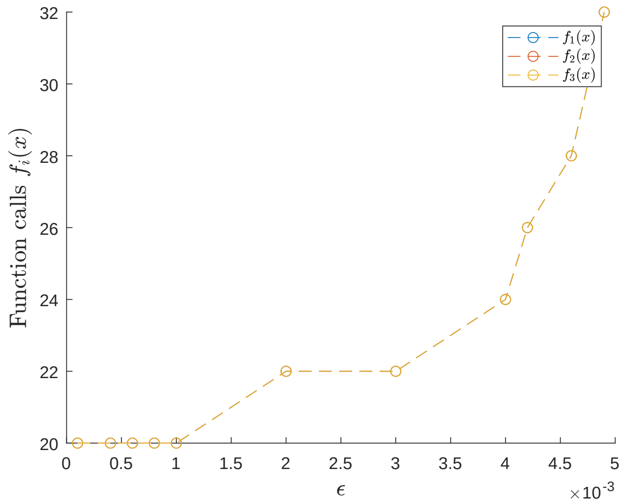
Εικόνα 1: Γραφική αναπαράσταση συναρτήσεων και των ελάχιστων σημείων τους

Βασικός άξονας των μεθόδων που θα ακολουθήσουν είναι ο περιορισμός του διαστήματος αναζήτησης $[\alpha,\beta]$ υπολογίζοντας την τιμή της f σε δύο εσωτερικά σημεία του. Αυτή είναι μια ιδιότητα των αυστηρών κυρτών συναρτήσεων. Έτσι για δύο σημεία $x1$ και $x2$, αν $f(x1) < f(x2)$ τότε το νέο διάστημα αναζήτησης θα είναι $[\alpha, x2]$ αλλιώς $[x1,\beta]$. Με ανάλογο τρόπο, επαναλαμβάνουμε το προηγούμενο σκεπτικό μέχρι να φτάσουμε στο επιθυμητό διάστημα που υποδηλώνει την ακρίβεια της προσέγγισης μας.

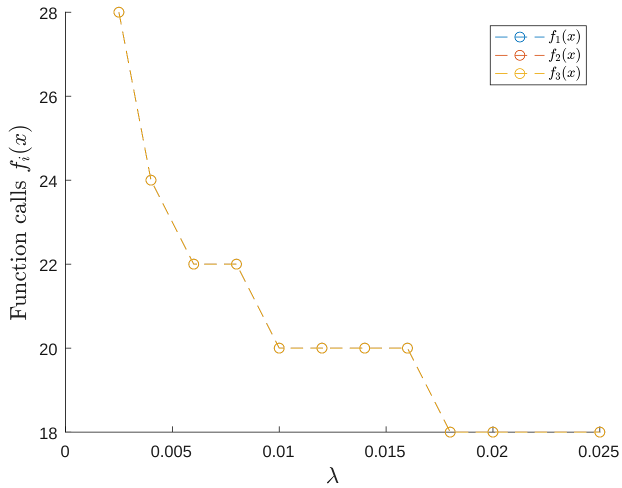
Θέμα 1 - Μέθοδος της Διχοτόμου

Για την μέθοδο της διχοτόμου, έχοντας μια αυστηρή κυρτή συνάρτηση $f(x)$, όπου $x \in [a,b]$, επιλέγουμε δύο σημεία $x1$ και $x2$ εντός του διαστήματος, όπου $x1 = \frac{\beta+\alpha}{2} - \epsilon$ και $x2 = \frac{\beta+\alpha}{2} + \epsilon$ με $\epsilon > 0$ θετική σταθερά. Η επιλογή αυτών των σημείων έγινε με γνώμονα την δημιουργία ίσου εύρους πιθανών διαστημάτων. Αξίζει να σημειωθεί ότι για ϵ, η τελική ακρίβεια του διαστήματος (l) δεν μπορεί να είναι μικρότερη απο $2e$ $(l > 2\epsilon)$. Η συνάρτηση που υλοποιεί την παραπάνω μέθοδο ονομάζεται `bisection()` και μπορεί να βρεθεί στο τέλος της αναφοράς.

Στη συνέχεια, θα παρουσιάσουμε κάποια γραφήματα για να μελετήσουμε την συμπεριφορά αυτής της μεθόδου.



(a) Μεταβολή του αριθμού κλήσεων της συνάρτησης f μεταβάλλοντας την σταθερά ϵ και σταθερό $l = 0.01$



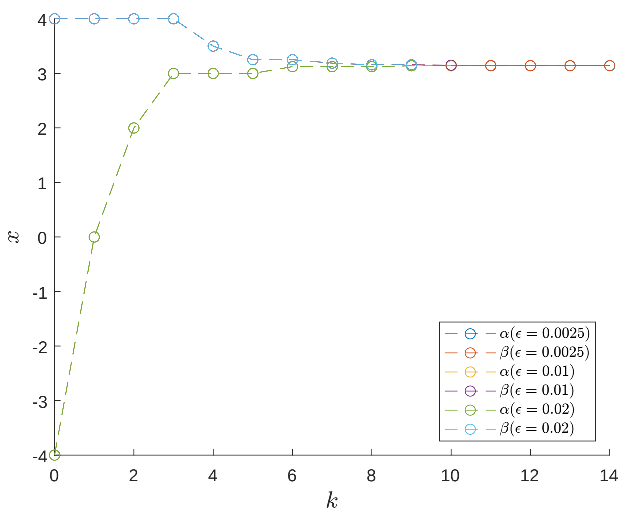
(b) Μεταβολή του αριθμού κλήσεων των συναρτήσεων $f_i(x)$ μεταβάλλοντας την σ ταθερά l και σταθερό $\epsilon = 0.001$

Εικόνα 2

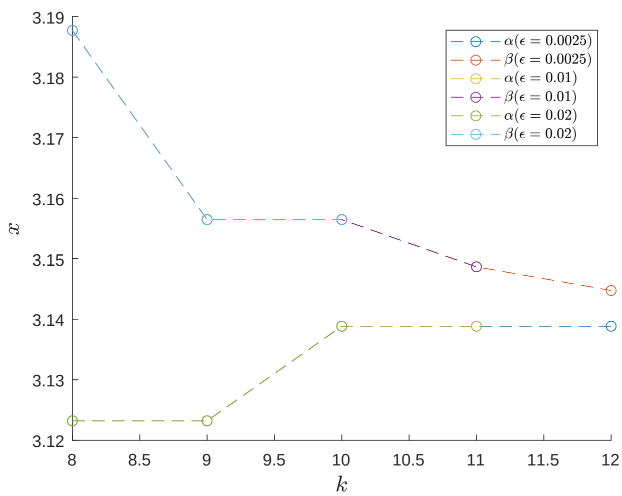
Παρατηρώντας την εικόνα 2, ο αριθμός των κλήσεων της αντικειμενικής συνάρτησης $f_i(x)$ αυξάνεται όταν μειώνουμε την σταθερά ϵ και όταν αυξάνουμε την σταθερά l. Αυτό φυσικά είναι αναμενόμενο καθώς μικρότερο ϵ και ως συνεπακόλουθο μεγαλύτερο l σημαίνει μεγαλύτερη ακρίβεια, άρα μικρότερο τελικό διάστημα και συνεπώς περισσότερες προσπάθειες διαμέρισης του διαστήματος.

'Οσον αφορά τον τύπο της συνάρτησης, δεν παρατηρείται κάποια μεταβολή και έχουμε όμοια γραφήματα για τις τρεις συναρτήσεις.

Στη συνέχεια παρουσιάζουμε τρία ζεύγη γραφημάτων (ένα για κάθε συνάρτηση) και μελετάμε την σύγκλιση των άκρων $[\alpha, \beta]$ ως προς τον αριθμό των επαναλήψεων k. Θεωρήσαμε σταθερό $\epsilon = 0.001$ και στο ίδιο γράφημα παρουσιάζουμε την εξέλιξη των άκρων για $l = 0.0025, 0.01, 0.02$. Για να διακρίνουμε την διαφορά ως προς την παραμετροποίηση του l, στο ζεύγος των γραφημάτων υπάρχει μια μεγενθυμένη εκδοχή.

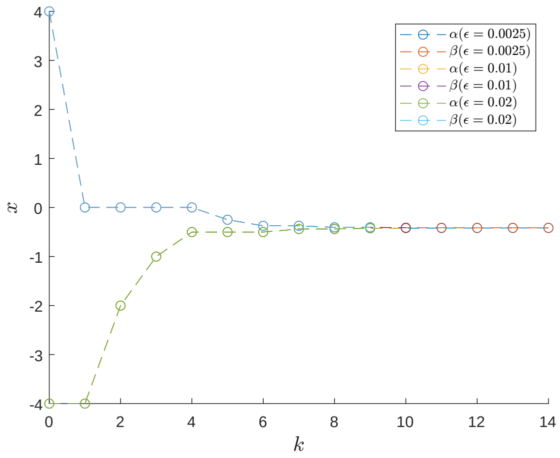


(a)

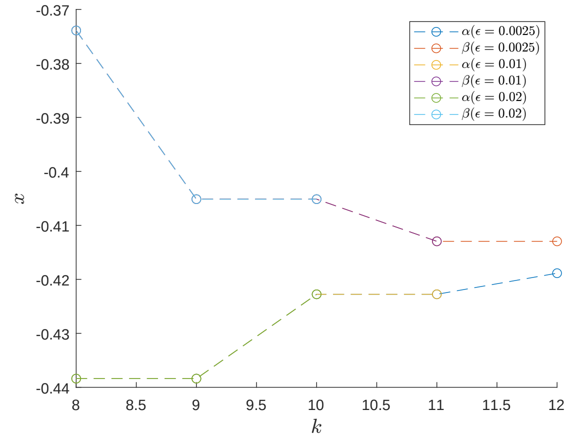


(b)

Εικόνα 3: Μεταβολή του διαστήματος αναζήτησης ως προς τον αριθμό των επαναλήψεων για $f_1(x)$

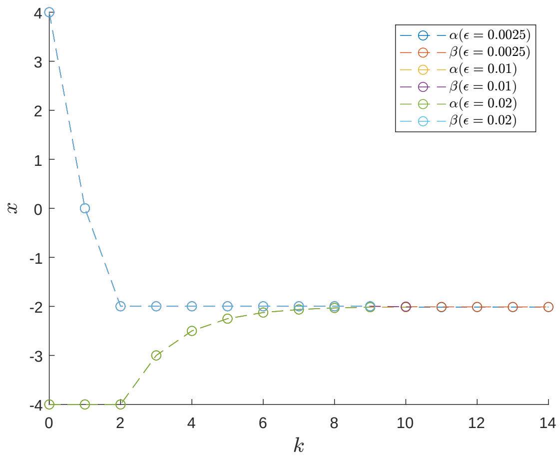


(a)

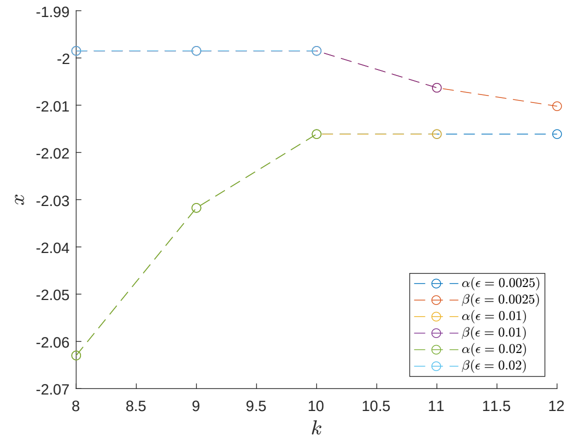


(b)

Εικόνα 4: Μεταβολή του διαστήματος αναζήτησης ως προς τον αριθμό των επαναλήψεων για $f_2(x)$



(a)

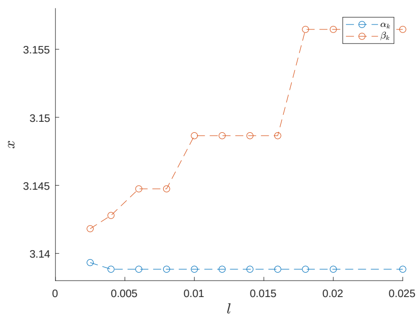


(b)

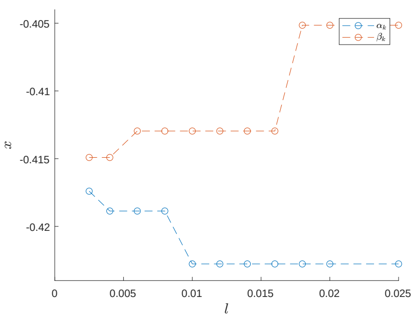
Εικόνα 5: Μεταβολή του διαστήματος αναζήτησης ως προς τον αριθμό των επαναλήψεων για $f_3(x)$

Όπως είναι αναμενόμενο, τα άκρα α, β συγκλίνουν όσο αυξάνουμε τον αριθμό των επαναλήψεων k. Παρατηρώντας την μεγενθυμένη εκδοχή των γραφημάτων μπορούμε να εντοπίσουμε την διαφορά ως προς την μεταβολή του l από την αλλαγή των χρωμάτων για τα α και β. Πιο συγκεκριμένα, βλέπουμε συνολικά ότι για κάθε συνάρτηση, τα άκρα α, β ακολουθούν την ίδια πορεία μέχρι να φτάσουν στο k που ικανοποιεί την προδιαγραφή l.

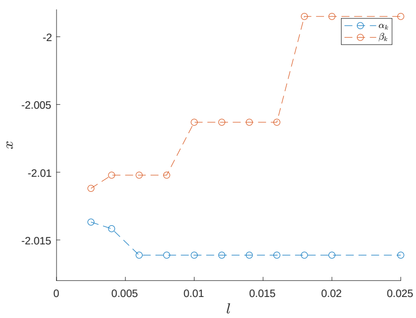
Παρακάτω, προσθέτουμε για κάθε l, το τελικό διάστημα $[a_k, b_k]$ μετά από k επαναλήψεις.



(a) $f_1(x)$



(b) $f_2(x)$



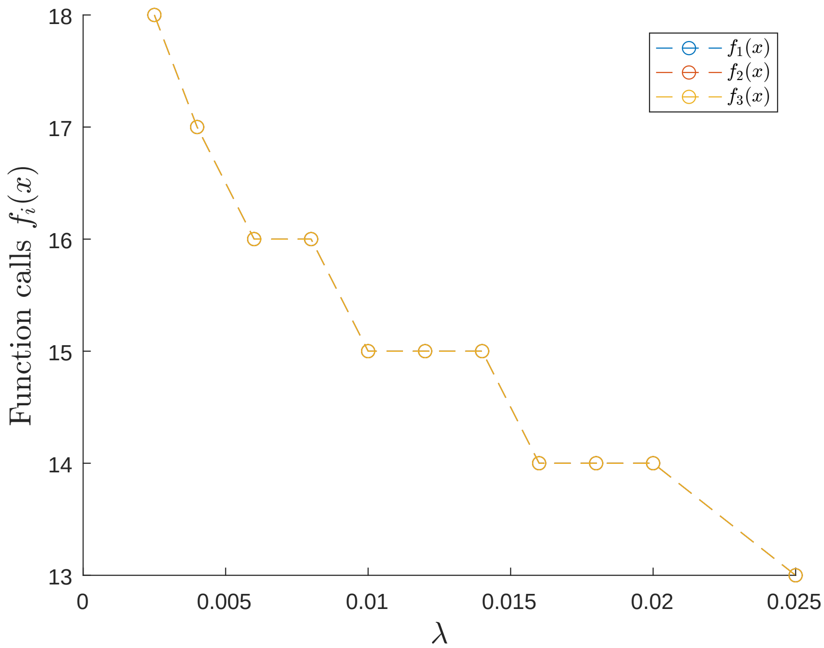
(c) $f_3(x)$

Εικόνα 6: Τελικό διάστημα $[a_k, b_k]$ ως προς μεταβολή του l

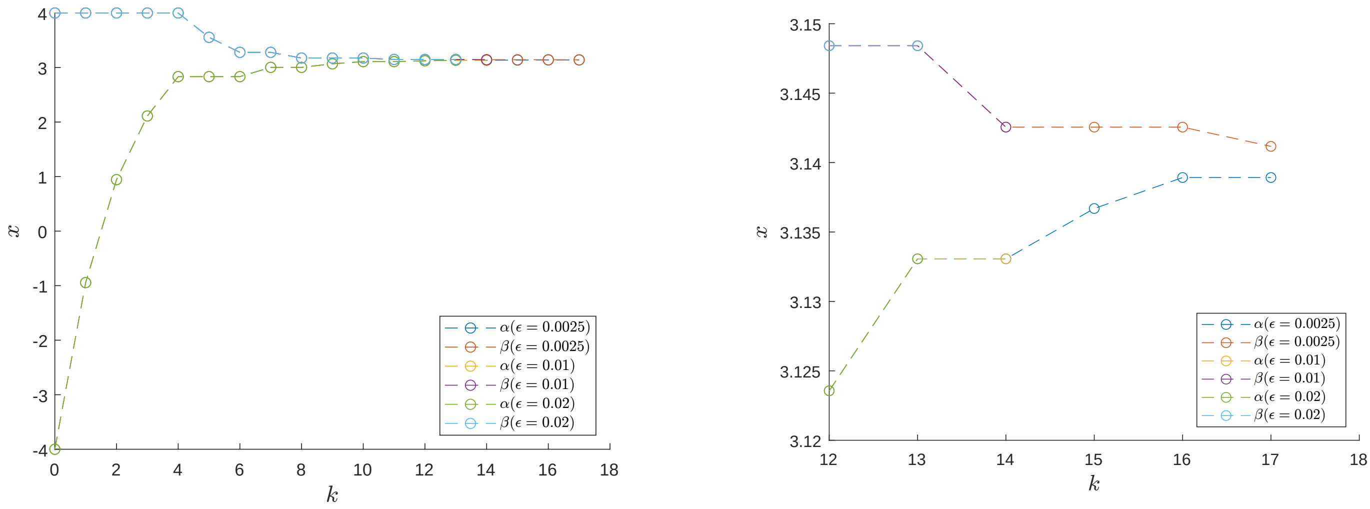
Θέμα 2 - Μέθοδος του Χρυσού Τομέα

Η μέθοδος του χρυσού τομέα, για την στρατηγική τοποθέτηση των εσωτερικών σημείων, είναι εμπνευσμένη από την χρυσή τομή. Έτσι έχουμε τον εξής υπολογισμό για το νέο διάστημα: $x1 = \alpha + (1-\gamma)(\beta - \alpha)$ και $x2 = \alpha + \gamma(\beta - \alpha)$, όπου $\gamma = \phi - 1 \approx 0.618$ και ϕ η χρυσή τομή.

Όμοια με την προηγούμενη ανάλυση έχουμε τα παρακάτω γραφήματα.

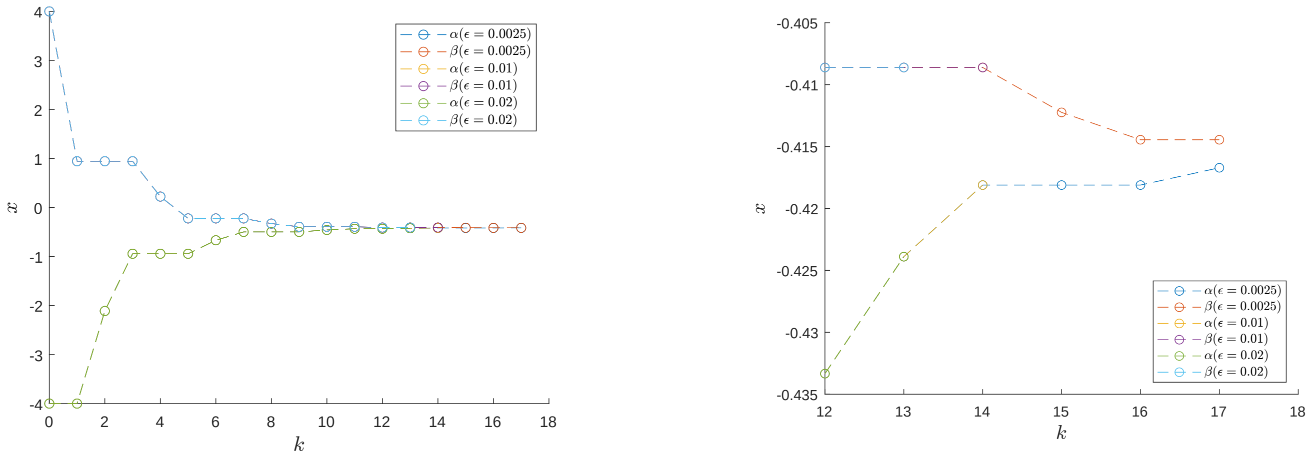


Εικόνα 7: Μεταβολή του αριθμού κλήσεων των συναρτήσεων $f_i(x)$ μεταβάλλοντας την σταθερά l και σταθερό $\epsilon = 0.001$



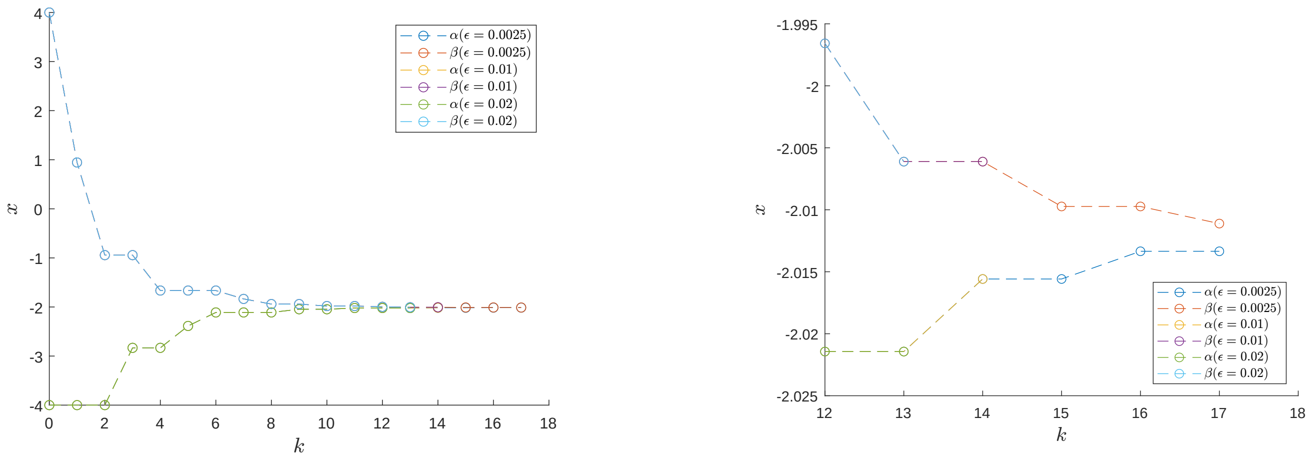
(a) (b)

Εικόνα 8: Μεταβολή του διαστήματος αναζήτησης ως προς τον αριθμό των επαναλήψεων για $f_1(x)$



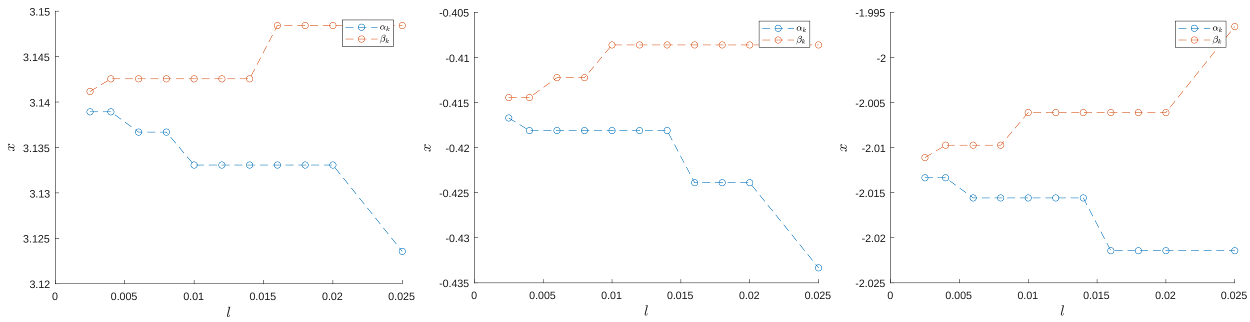
(a) (b)

Εικόνα 9: Μεταβολή του διαστήματος αναζήτησης ως προς τον αριθμό των επαναλήψεων για $f_2(x)$



(a) (b)

Εικόνα 10: Μεταβολή του διαστήματος αναζήτησης ως προς τον αριθμό των επαναλήψεων για $f_3(x)$



(a) $f_1(x)$ (b) $f_2(x)$ (c) $f_3(x)$

Εικόνα 11: Τελικό διάστημα $[a_k, b_k]$ ως προς μεταβολή του l

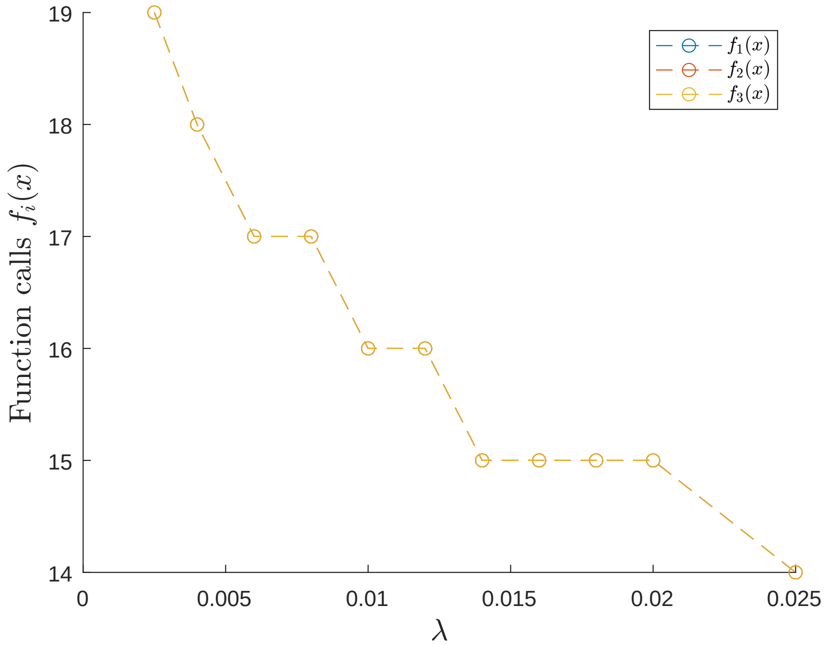
Σε σχέση με την προηγούμενη μέθοδο της διχοτόμου, μπορούμε να συμπεράνουμε ότι ο αριθμός των κλήσεων της αντικειμενικής συνάρτησης έχει μειωθεί αλλά ο αριθμός των επαναλήψεων k έχει αυξηθεί.

Θέμα 3 - Μέθοδος Fibonacci

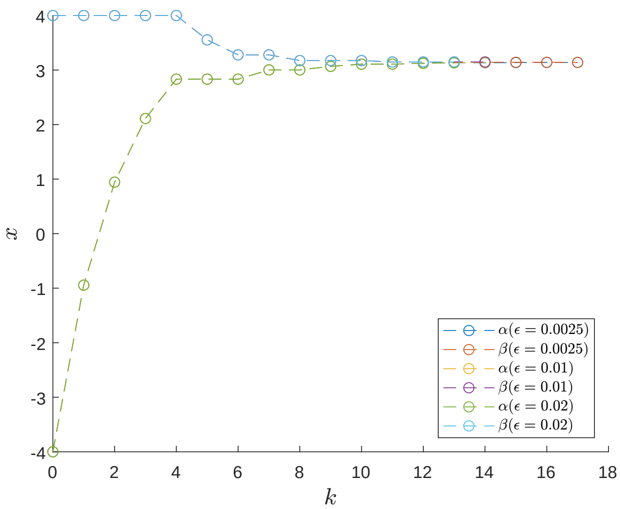
Η επιλογή των εσωτερικών σημείων της μεθόδου Fibonacci βασίζεται στην ακολουθία Fibonacci και συσχετίζεται με την μέθοδο του χρυσού τομέα. Σημαντική διαφορά με τις προηγούμενες περιπτώσεις είναι ο προϋπολογισμός των επαναλήψεων που θα χρειαστούν για να φτάσουμε στο επιθυμητό διάστημα αναζήτησης.

Έτσι λοιπόν βρίσκουμε τον μικρότερο αριθμό n που ικανοποιεί την σχέση $F_n > \frac{b-a}{l}$ και έχουμε $x1 = \alpha + \frac{F_{n-2-k}}{F_{n-k}}(\beta - \alpha)$ και $x2 = \alpha + \frac{F_{n-1-k}}{F_{n-k}}(\beta - \alpha)$ όπου $k \in [0, n-2]$ και F_n ακολουθία fibonacci με $F_0 = F_1 = 1$.

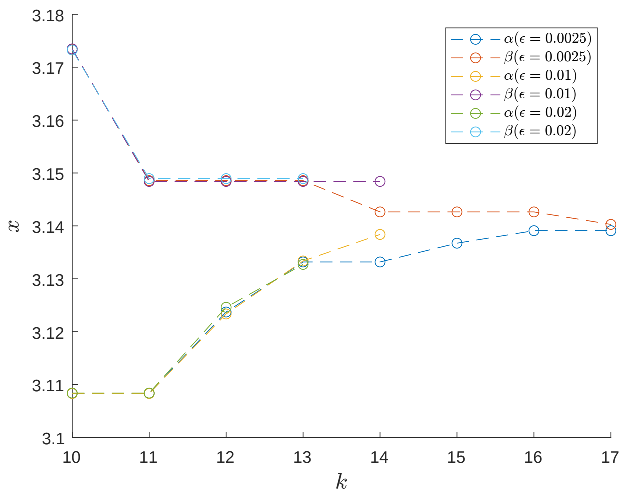
Η συχέτιση μεταξύ Fibonacci και χρυσής τομής φαίνεται απο την σχέση $lim_{n\to\infty}\frac{F_{n-1}}{F_n} = 0.618$. Με άλλα λόγια για μεγάλο n, οι δύο αυτές μέθοδοι συμπίπτουν.



Εικόνα 12: Μεταβολή του αριθμού κλήσεων των συναρτήσεων $f_i(x)$ μεταβάλλοντας την σταθερά l και σταθερό $\epsilon = 0.001$

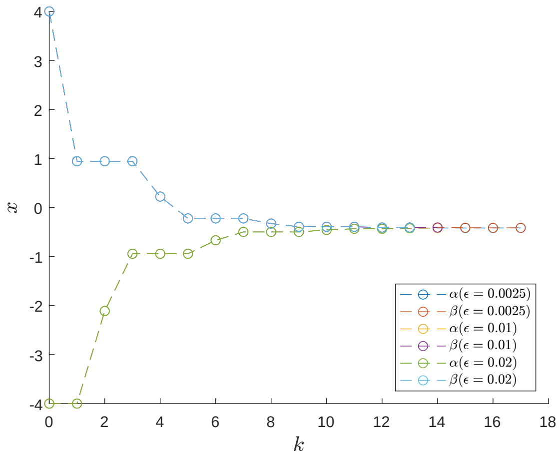


(a)

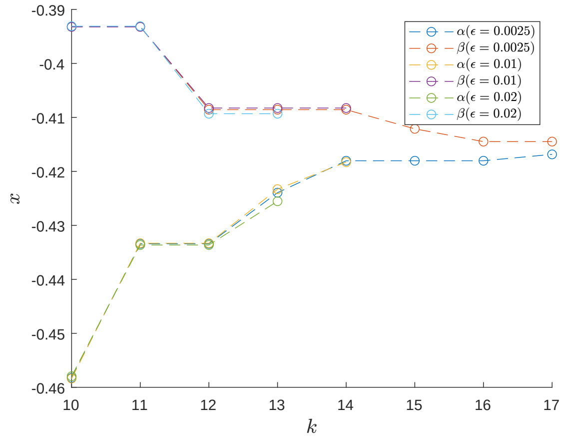


(b)

Εικόνα 13: Μεταβολή του διαστήματος αναζήτησης ως προς τον αριθμό των επαναλήψεων για $f_1(x)$

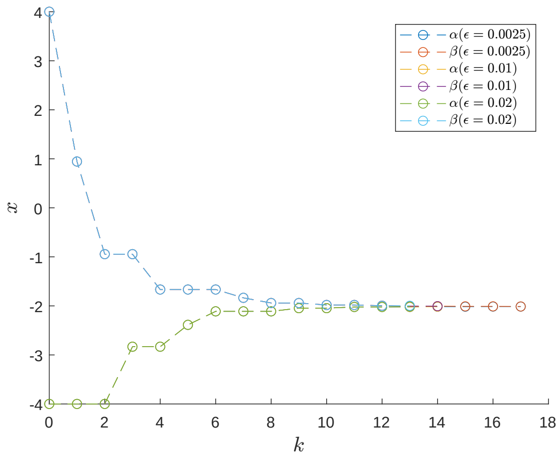


(a)

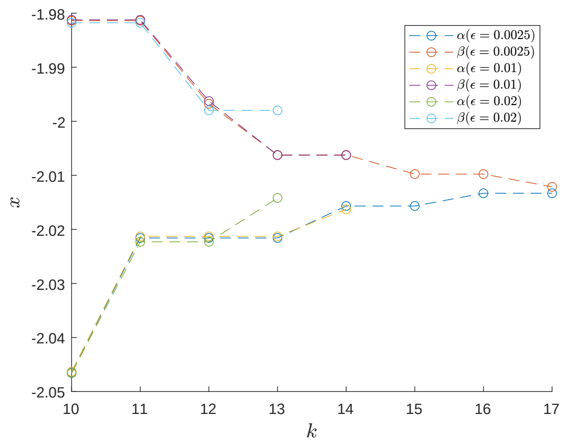


(b)

Εικόνα 14: Μεταβολή του διαστήματος αναζήτησης ως προς τον αριθμό των επαναλήψεων για $f_2(x)$

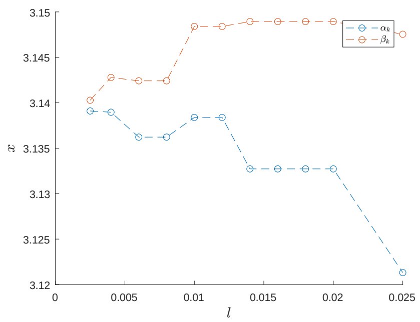


(a)

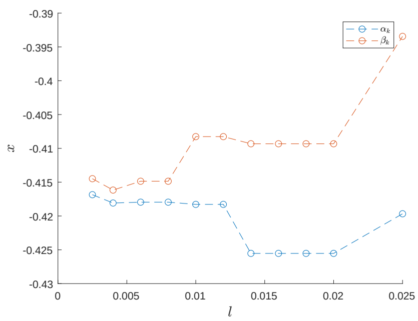


(b)

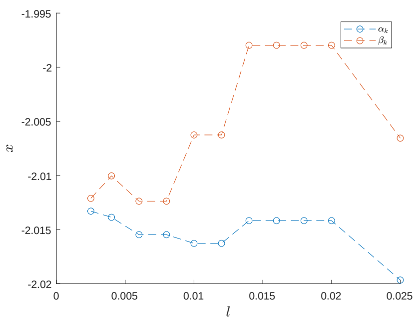
Εικόνα 15: Μεταβολή του διαστήματος αναζήτησης ως προς τον αριθμό των επαναλήψεων για $f_3(x)$



(a) $f_1(x)$



(b) $f_2(x)$



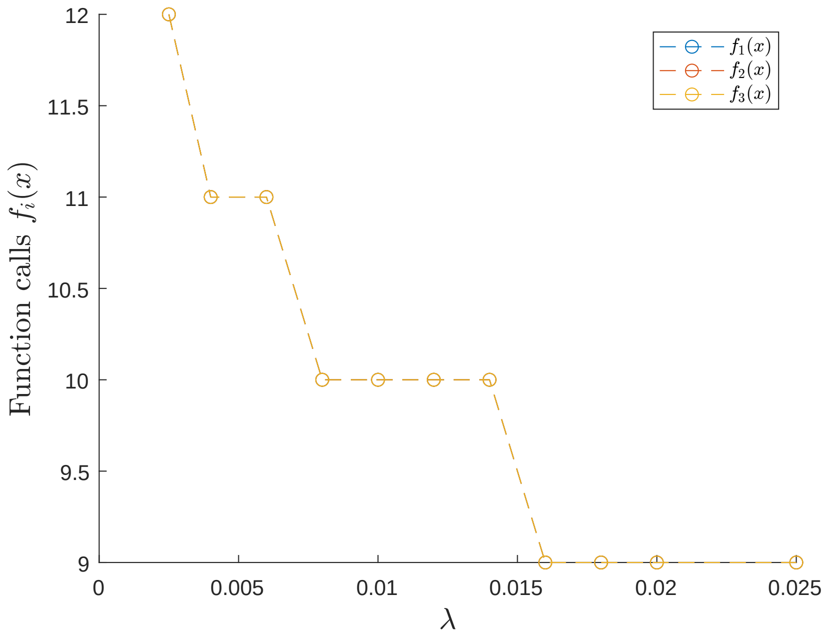
(c) $f_3(x)$

Εικόνα 16: Τελικό διάστημα $[a_k, b_k]$ ως προς μεταβολή του l

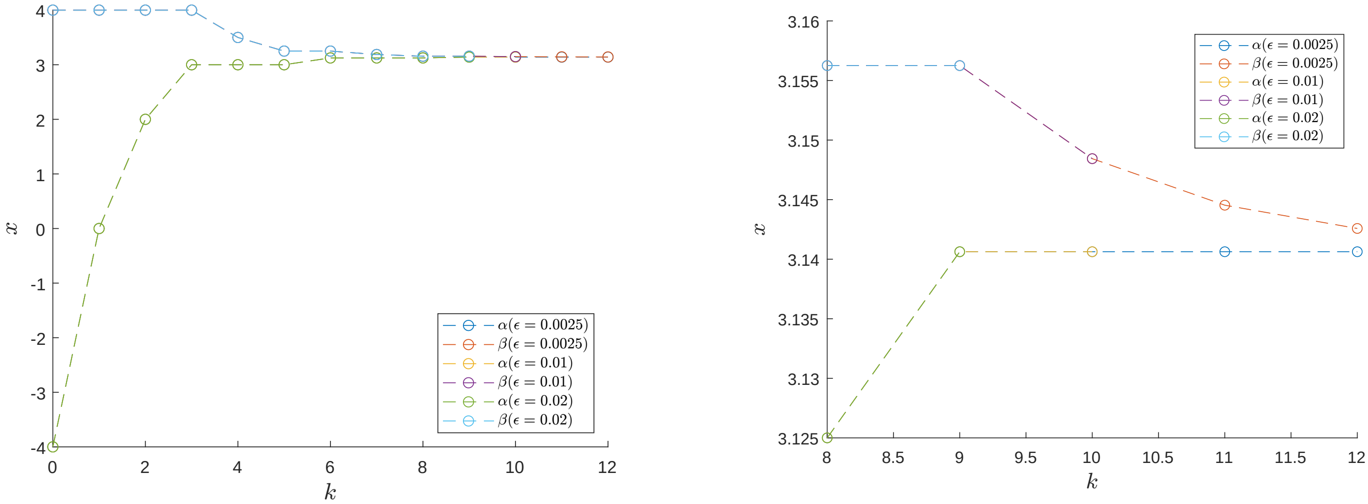
Παρατηρούμε ότι σε αυτήν την περίπτωση μεταβάλλοντας το l, τα διαστήματα των μεγαλύτερων l δεν αποτελούν υποσύνολο των μικρότερων και υπάρχει μικρή απόκλιση ως προς τον τρόπο εξέλιξης των διαστημάτων αναζήτησης.

Θέμα 4 - Μέθοδος της Διχοτόμου με χρήση παραγώγου

Σε αυτό το θέμα, η συγκεκριμένη μέθοδος είναι η μόνη που βασίζεται στην παραγώγιση της συνάρτησης, η οποία ανάλογα το πρόσημο της προσδιορίζει το επόμενο μικρότερο διάστημα αναζήτησης. Ο αριθμός των επαναλήψεων υπολογίζεται εκ των προτέρων και επιλέγεται το μεγαλύτερο n που ικανοποιεί την σχέση $(\frac{1}{2})^n \leq \frac{l}{\beta-\alpha}$.

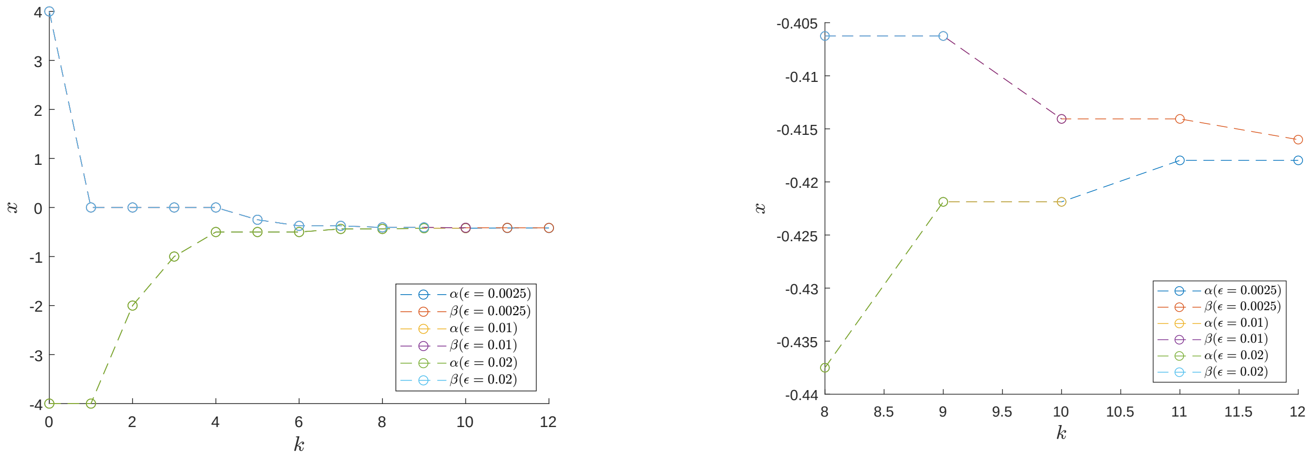


Εικόνα 17: Μεταβολή του αριθμού κλήσεων των συναρτήσεων $f_i(x)$ μεταβάλλοντας την σταθερά l και σταθερό $\epsilon = 0.001$



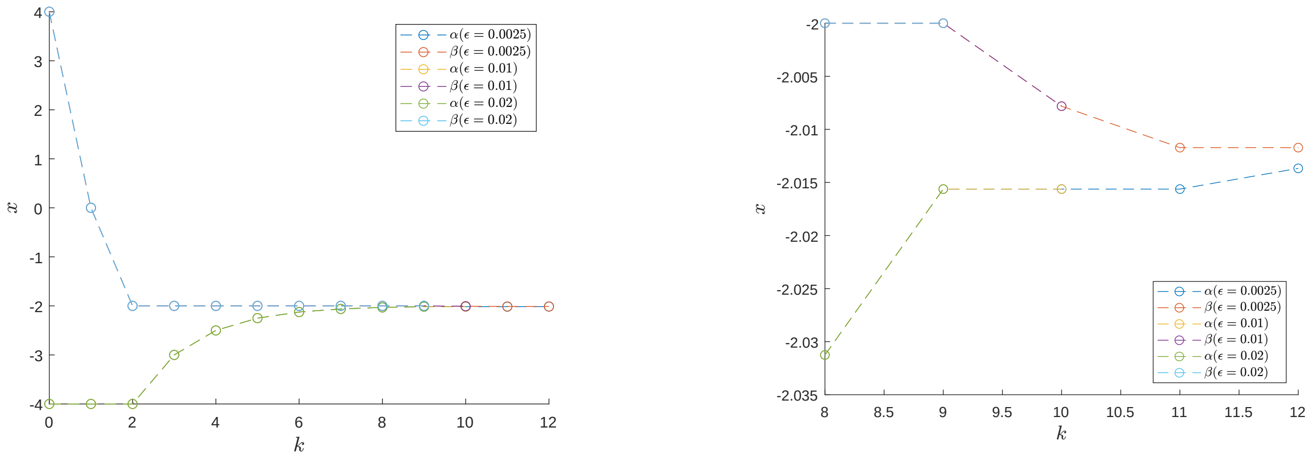
(a) (b)

Εικόνα 18: Μεταβολή του διαστήματος αναζήτησης ως προς τον αριθμό των επαναλήψεων για $f_1(x)$



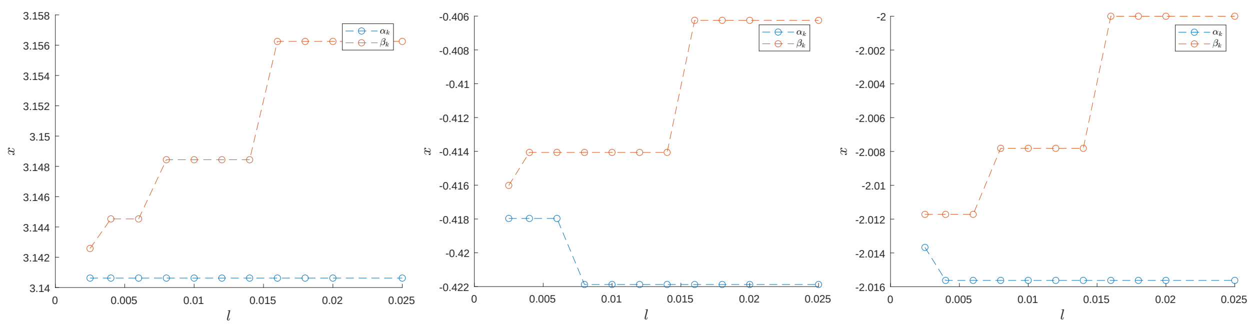
(a) (b)

Εικόνα 19: Μεταβολή του διαστήματος αναζήτησης ως προς τον αριθμό των επαναλήψεων για $f_2(x)$



(a) (b)

Εικόνα 20: Μεταβολή του διαστήματος αναζήτησης ως προς τον αριθμό των επαναλήψεων για $f_3(x)$



(a) $f_1(x)$ (b) $f_2(x)$ (c) $f_3(x)$

Εικόνα 21: Τελικό διάστημα $[a_k, b_k]$ ως προς μεταβολή του l

Παρατηρούμε ότι με αυτήν την μέθοδο, ο αριθμός των κλήσεων της $\dot{f}$ είναι ο μικρότερος απο όλες τις προηγούμενες περιπτώσεις.

Matlab κώδικας

Για όλα τα παραπάνω παρατίθεται ο κώδικας Matlab.

```
%% MINIMIZATION PROBLEM FOR CONVEX FUNCTIONS SINGLE VARIABLE
clc
clear all

% FUNCTIONS
f1 = @(x) (x-3)^2 + (sin(x+3))^2;
f2 = @(x) (x-1) * cos(0.5 * x) + x^2;
f3 = @(x) (x+2)^2 + exp(x-2) * sin(x+3);
numFuncs = 3;

intervalStart = [-4 4];

syms x;
figure
hold on
h1 = fplot(f1(x),intervalStart);
g = diff(f1,x); solve(g == 0, x); extrema = vpa(ans);
plot(extrema, subs(sym(f1),extrema), '*','color','red')

h2 = fplot(f2(x),intervalStart);
g = diff(f2,x); solve(g == 0, x); extrema = vpa(ans);
plot(extrema, subs(sym(f2),extrema), '*','color','red')

h3 = fplot(f3(x),intervalStart);
g = diff(f3,x); solve(g == 0, x); extrema = vpa(ans);
plot(extrema, subs(sym(f3),extrema), '*','color','red')

legend([h1 h2 h3],'$f_1(x)$','$f_2(x)$','$f_3(x)$','Interpreter','latex');
exportgraphics(gcf,'functions_symbolic.pdf','ContentType','vector')

%% BISECTION METHOD
% STABLE ACCURACY (lambda) VARIABLE STEP (epsilon)
intervalAccuracy = 0.01;
intervalStep = [0.0001 0.0004 0.0006 0.0008 0.001 0.002 0.003 0.004 0.0042 ...
    0.0046 0.0049];
ends = cell(numFuncs,numel(intervalStep));
countFunctionCalls = cell(numFuncs,numel(intervalStep));

funcs=cell(numFuncs,1);
funcs{1} = f1; funcs{2} = f2; funcs{3} = f3;
for idFunc=1:numel(funcs)
    for i=1:numel(intervalStep)
        [endPoints,count] = ...
            bisection(intervalStart,intervalStep(i),intervalAccuracy,funcs{idFunc});
        ends(idFunc,i) = {endPoints};
        countFunctionCalls(idFunc,i) = {count};
    end
end

%% BISECTION METHOD
% STABLE STEP (epsilon) VARIABLE ACCURACY (lambda)
intervalStep = 0.001;
intervalAccuracy = [0.0025 0.004 0.006 0.008 0.01 0.012 0.014 0.016 0.018 0.02 ...
    0.025];
ends = cell(numFuncs,numel(intervalAccuracy));
countFunctionCalls = cell(numFuncs,numel(intervalAccuracy));

```

```
funcs=cell(numFuncs,1);
funcs{1} = f1; funcs{2} = f2; funcs{3} = f3;
for idFunc=1:numel(funcs)
    for i=1:numel(intervalAccuracy)
        [endPoints,count] = ...
            bisection(intervalStart,intervalStep,intervalAccuracy(i),funcs{idFunc});
        ends(idFunc,i) = {endPoints};
        countFunctionCalls(idFunc,i) = {count};
    end
end

%% GOLDEN SECTOR METHOD
intervalAccuracy = [0.0025 0.004 0.006 0.008 0.01 0.012 0.014 0.016 0.018 0.02 ...
    0.025];
ends = cell(numFuncs,numel(intervalAccuracy));
countFunctionCalls = cell(numFuncs,numel(intervalAccuracy));

funcs=cell(numFuncs,1);
funcs{1} = f1; funcs{2} = f2; funcs{3} = f3;
for idFunc=1:numel(funcs)
    for i=1:numel(intervalAccuracy)
        [endPoints,count] = ...
            goldenSector(intervalStart,intervalAccuracy(i),funcs{idFunc});
        ends(idFunc,i) = {endPoints};
        countFunctionCalls(idFunc,i) = {count};
    end
end

%% FIBONACCI METHOD
intervalStep = 0.001;
intervalAccuracy = [0.0025 0.004 0.006 0.008 0.01 0.012 0.014 0.016 0.018 0.02 ...
    0.025];
ends = cell(numFuncs,numel(intervalAccuracy));
countFunctionCalls = cell(3,numel(intervalAccuracy));

funcs=cell(numFuncs,1);
funcs{1} = f1; funcs{2} = f2; funcs{3} = f3;
for idFunc=1:numel(funcs)
    for i=1:numel(intervalAccuracy)
        [endPoints,count] = ...
            fib(intervalStart,intervalStep,intervalAccuracy(i),funcs{idFunc});
        ends(idFunc,i) = {endPoints};
        countFunctionCalls(idFunc,i) = {count};
    end
end

%% BISECTION DERIVATIVE METHOD
intervalAccuracy = [0.0025 0.004 0.006 0.008 0.01 0.012 0.014 0.016 0.018 0.02 ...
    0.025];
ends = cell(numFuncs,numel(intervalAccuracy));
countFunctionCalls = cell(numFuncs,numel(intervalAccuracy));

funcs=cell(numFuncs,1);
funcs{1} = f1; funcs{2} = f2; funcs{3} = f3;
for idFunc=1:numel(funcs)
    for i=1:numel(intervalAccuracy)
        [endPoints,count] = ...
            bisectionDerivative(intervalStart,intervalAccuracy(i),funcs{idFunc});
        ends(idFunc,i) = {endPoints};
        countFunctionCalls(idFunc,i) = {count};
    end
end

```

```
%% PLOTS
% varying epsilon (used for bisection method)
figure
hold on
plot(intervalStep,cell2mat(countFunctionCalls(1,:)),'--o')
plot(intervalStep,cell2mat(countFunctionCalls(2,:)),'--o')
plot(intervalStep,cell2mat(countFunctionCalls(3,:)),'--o')
xlabel('$\epsilon$','fontsize',14,'interpreter','latex')
ylabel('Function calls $f_i(x)$','fontsize',14,'interpreter','latex')

legend('$f_1(x)$','$f_2(x)$','$f_3(x)$','Interpreter','latex');
%exportgraphics(gcf,'bisection_epsilon.pdf','ContentType','vector')

%% varying end points with stable lambda
figure
hold on
intervalStepIndex = 11;
for i = 1:3
pairs = cell2mat(ends(i,intervalStepIndex));
[k,] = size(pairs);
plot(1:k,pairs(:,1),'--o')
plot(1:k,pairs(:,2),'--o')
xlabel('$k$','fontsize',14,'interpreter','latex')
ylabel('Function calls $f_1(x)$','fontsize',14,'interpreter','latex')
end

%% varying lambda
figure
hold on
plot(intervalAccuracy,cell2mat(countFunctionCalls(1,:)),'--o')
plot(intervalAccuracy,cell2mat(countFunctionCalls(2,:)),'--o')
plot(intervalAccuracy,cell2mat(countFunctionCalls(3,:)),'--o')
xlabel('$\lambda$','fontsize',14,'interpreter','latex')
ylabel('Function calls $f_i(x)$','fontsize',14,'interpreter','latex')

legend('$f_1(x)$','$f_2(x)$','$f_3(x)$','Interpreter','latex');
exportgraphics(gcf,'bisection_deriv_lambda.pdf','ContentType','vector')

%% varying end points with stable epsilon with respect to iterations k
figure
hold on
for i=[1 5 10]
    pairs = cell2mat(ends(3,i));
    [k,] = size(pairs);
    plot(0:k-1,pairs(:,1),'--o')
    plot(0:k-1,pairs(:,2),'--o')
    xlabel('$k$','fontsize',14,'interpreter','latex')
    ylabel('$x$','fontsize',14,'interpreter','latex')
end
%xlim([8 12])
legend('$\alpha (\epsilon=0.0025)$','$\beta (\epsilon=0.0025)$',...
    '$\alpha (\epsilon=0.01)$','$\beta (\epsilon=0.01)$','$\alpha ... (\epsilon=0.02)$','$\beta (\epsilon=0.02)$',...
    'Interpreter','latex');
exportgraphics(gcf,'bisection_deriv_endpoints_f3.pdf','ContentType','vector')
%exportgraphics(gcf,'bisection_deriv_endpoints_f3_zoom.pdf','ContentType','vector')

%% varying end points with stable epsilon with respect to lambda
figure
count = 1;
for i=1:numel(intervalAccuracy)
    pairs = cell2mat(ends(3,i));
    finalA(count) = pairs(end,1);
```

```matlab
    finalB(count) = pairs(end,2);
    count = count+1;
end
hold on
plot(intervalAccuracy,finalA,'--o')
plot(intervalAccuracy,finalB,'--o')
xlabel('$l$','fontsize',14,'interpreter','latex')
ylabel('$x$','fontsize',14,'interpreter','latex')
legend('$\alpha_k$','$\beta_k$','interpreter','latex')
%exportgraphics(gcf,'bisection_deriv_endpoints_f3_lambda.pdf','ContentType','vector')

function [ends,countFunctionCalls] = ...
    bisection(intervalStart,intervalStep,intervalAccuracy,f)
    a(1) = intervalStart(1);
    b(1) = intervalStart(2);
    countFunctionCalls = 0;
    i = 1;
    while b(i)-a(i) >= intervalAccuracy
        midPoint = (a(i)+b(i))/2;
        x1 = midPoint - intervalStep;
        x2 = midPoint + intervalStep;
        if f(x1) < f(x2)
            a(i+1) = a(i);
            b(i+1) = x2;
        else
            a(i+1) = x1;
            b(i+1) = b(i);
        end
        countFunctionCalls = countFunctionCalls+2;
        i = i+1;
    end

    ends(:,1) = a;
    ends(:,2) = b;
end

function [ends,countFunctionCalls] = ...
    goldenSector(intervalStart,intervalAccuracy,f)
    % init
    gamma = 0.618;
    a(1) = intervalStart(1);
    b(1) = intervalStart(2);
    nextX1 = @(a,b,gamma) a + (1-gamma)*(b-a);
    nextX2 = @(a,b,gamma) a + gamma*(b-a);
    x1 = nextX1(a(1),b(1),gamma);
    x2 = nextX2(a(1),b(1),gamma);
    y1 = f(x1);
    y2 = f(x2);

    i = 1;
    countFunctionCalls = 2;
    while (b(i)-a(i)) >= intervalAccuracy
        if y1 < y2
            a(i+1) = a(i);
            b(i+1) = x2;
            x2 = x1;
            x1 = nextX1(a(i+1),b(i+1),gamma);
            y2 = y1;
            y1 = f(x1);
        else
            a(i+1) = x1;
            b(i+1) = b(i);
            x1 = x2;
```

```
            x2 = nextX2(a(i+1),b(i+1),gamma);
            y1 = y2;
            y2 = f(x2);
        end
        countFunctionCalls = countFunctionCalls+1;
        i = i+1;
    end
    % remove last redundant call
    countFunctionCalls = countFunctionCalls-1;

    ends(:,1) = a;
    ends(:,2) = b;
end

function [ends,countFunctionCalls] = ...
    fib(intervalStart,intervalStep,intervalAccuracy,f)
    % init
    a(1) = intervalStart(1);
    b(1) = intervalStart(2);

    % find number of iterations
    %fib(1) = 1; fib(2) = 1;
    n = 1;
    while fibonacci(n) <= (b(1)-a(1))/intervalAccuracy
       n = n + 1;
    end
    n = n+1;
    %fprintf("num iterations: %d, ...
        ,%f,%0.2f\n",n,intervalAccuracy,(b(1)-a(1))/intervalAccuracy)

    nextX = @(a,b,ratio) a + ratio*(b-a);
    x1 = nextX(a(1),b(1),fibonacci(n-2)/fibonacci(n));
    x2 = nextX(a(1),b(1),fibonacci(n-1)/fibonacci(n));
    y1 = f(x1);
    y2 = f(x2);

    i = 1;
    countFunctionCalls = 2;
    while i <= n-3
        if y1 < y2
            if i == (n-3)
                a(i+1) = a(i);
                b(i+1) = x1;
            else
                a(i+1) = a(i);
                b(i+1) = x2;
                x2 = x1;
                x1 = nextX(a(i+1),b(i+1),fibonacci(n-2-i)/fibonacci(n-i));
                y2 = y1;
                y1 = f(x1);
            end
        else
            if i == (n-3)
                a(i+1) = x1;
                b(i+1) = b(i);
            else
                a(i+1) = x1;
                b(i+1) = b(i);
                x1 = x2;
                x2 = nextX(a(i+1),b(i+1),fibonacci(n-1-i)/fibonacci(n-i));
                y1 = y2;
                y2 = f(x2);
            end
```

```
            end
            countFunctionCalls = countFunctionCalls+1;
            i = i+1;
            if i == (n-3)
                x2 = x1 + intervalStep;
                y2 = f(x2);
            end
        end

        ends(:,1) = a;
        ends(:,2) = b;
    end

    function [ends,countFunctionCalls] = ...
        bisectionDerivative(intervalStart,intervalAccuracy,f)
        a(1) = intervalStart(1);
        b(1) = intervalStart(2);

        % derivative
        df = matlabFunction(diff(sym(f)));

        % find number of iterations
        n = 1;
        while((0.5)^(n) > intervalAccuracy/(b(1)-a(1)))
             n = n+1;
        end
        %fprintf("%0.2f,%d\n",intervalAccuracy/(b(1)-a(1)),n)

        countFunctionCalls = 0;
        i = 1;
        while i <= n
            x = (a(i)+b(i))/2;
            dx = df(x);
            countFunctionCalls = countFunctionCalls+1;
            if dx == 0
                a(i+1) = x;
                b(i+1) = x;
                break
            elseif dx < 0
                a(i+1) = x;
                b(i+1) = b(i);
            elseif dx > 0
                a(i+1) = a(i);
                b(i+1) = x;
            end
            i = i+1;
        end

        ends(:,1) = a;
        ends(:,2) = b;
    end
```