Τεχνικές Βελτιστοποίησης

Ελαχιστοποίηση κυρτής συνάρτησης μιας μεταβλητής σε δοσμένο διάστημα

Θεόδωρος Κατζάλης
AEM: 9282
`katzalis@auth.gr`

17/11/2021

Περιεχόμενα

Εισαγωγή 2

Θέμα 1 - Μέθοδος της Διχοτόμου 2

Θέμα 2 - Μέθοδος του Χρυσού Τομέα 4

Θέμα 3 - Μέθοδος Fibonacci 4

Θέμα 4 - Μέθοδος της Διχοτόμου με χρήση παραγώγου 4

Matlab κώδικας 5

Εισαγωγή

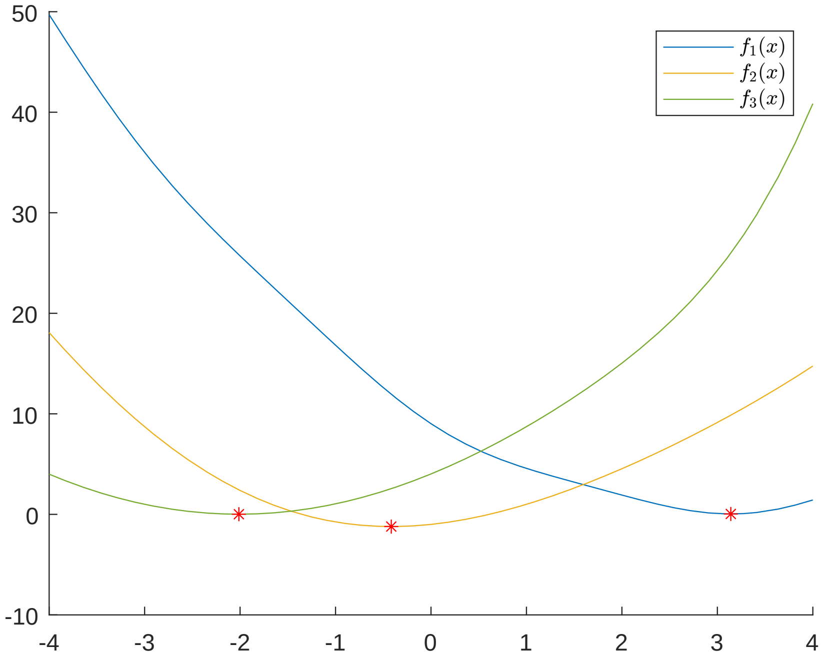
Σκοπός αυτής της εργασίας είναι η μελέτη διάφορων μεθόδων ελαχιστοποίησης κυρτής συνάρτησης $f(x)$ σε δοσμένο διάστημα $[\alpha, \beta]$. Οι συναρτήσεις που θα μελετηθούν με αρχικό διάστημα $[-4, 4]$ είναι οι εξής:

$$f_1(x) = (x-3)^2 + sin^2(x+3) \tag{1}$$

$$f_2(x) = (x-1)cos(\frac{1}{2}x) + x^2 \tag{2}$$

$$f_3(x) = (x+2)^2 + e^{x-2}sin(x+3) \tag{3}$$

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



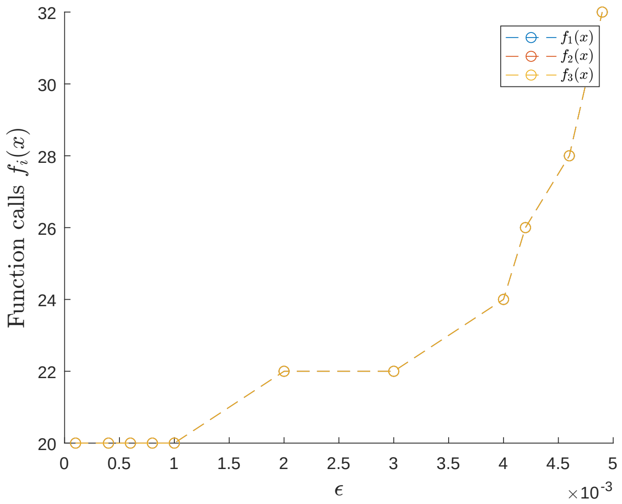
Εικόνα 1: Γραφική αναπαράσταση συναρτήσεων και των ελάχιστων σημείων τους

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.

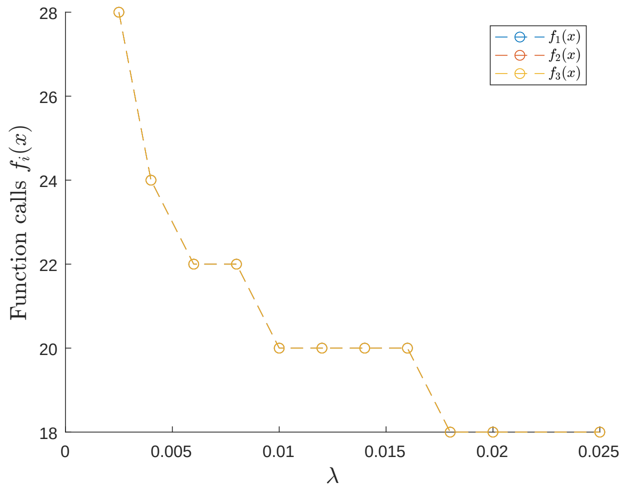
Θέμα 1 - Μέθοδος της Διχοτόμου

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 $l = 0.01$



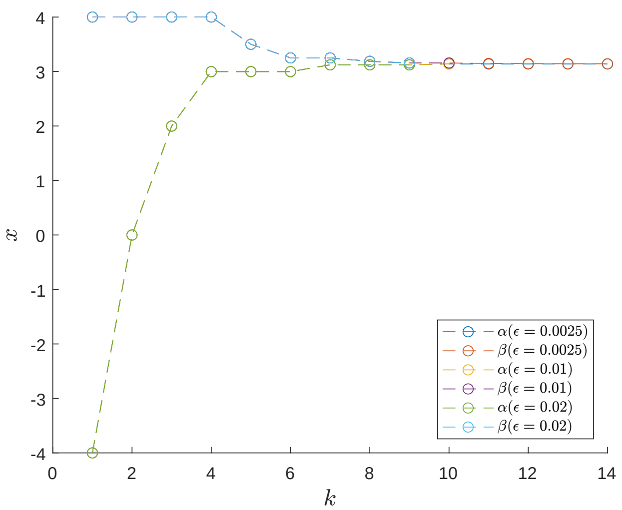
(b) Μεταβολή του αριθμού κλήσεων της συνάρτησης f μεταβάλλοντας την σταθερά l και σταθερό $\epsilon = 0.001$

Εικόνα 2

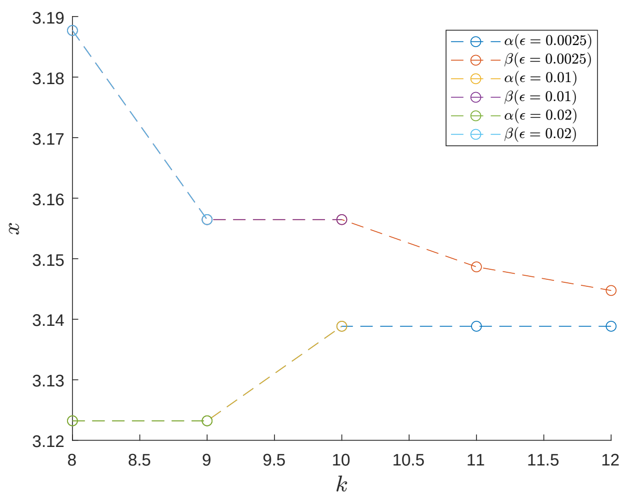
Παρατηρώντας την εικόνα 2, ο αριθμός των κλήσεων της αντικειμενικής συνάρτησης $f_i(x)$ αυξάνεται όταν μειώνουμε την σταθερά ϵ και όταν αυξάνουμε την σταθερά l. Αυτό φυσικά είναι αναμενόμενο καθώς μικρότερο ϵ και ως συνεπακόλουθο μεγαλύτερο l σημαίνει μεγαλύτερη ακρίβεια, άρα μικρότερο τελικό διάστημα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

Όσον 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 ως προς το l, 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.

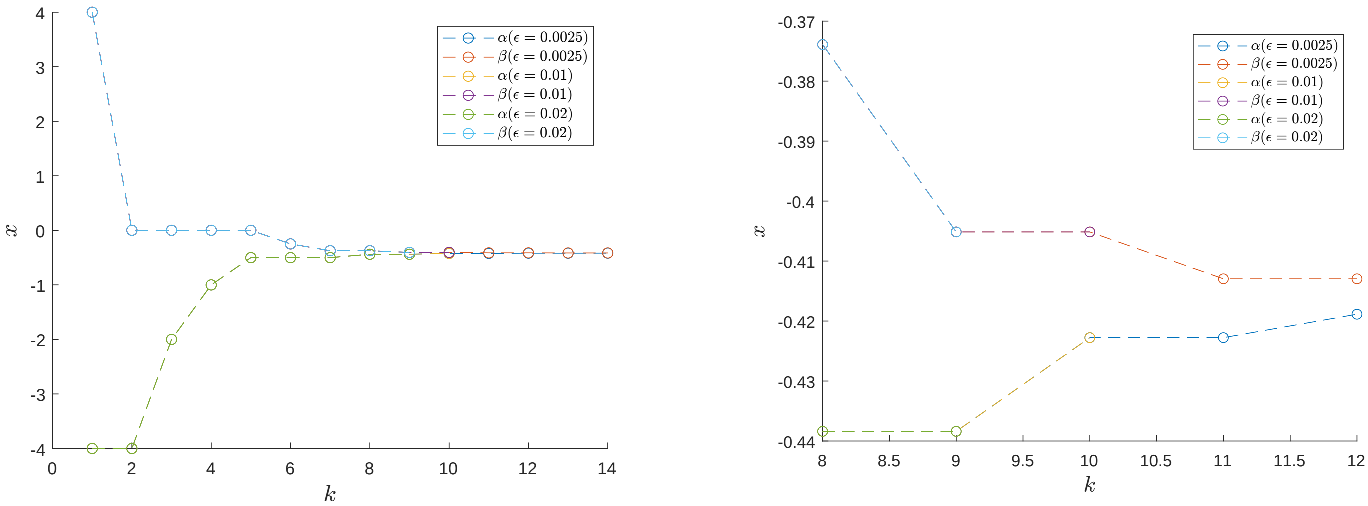


(a)



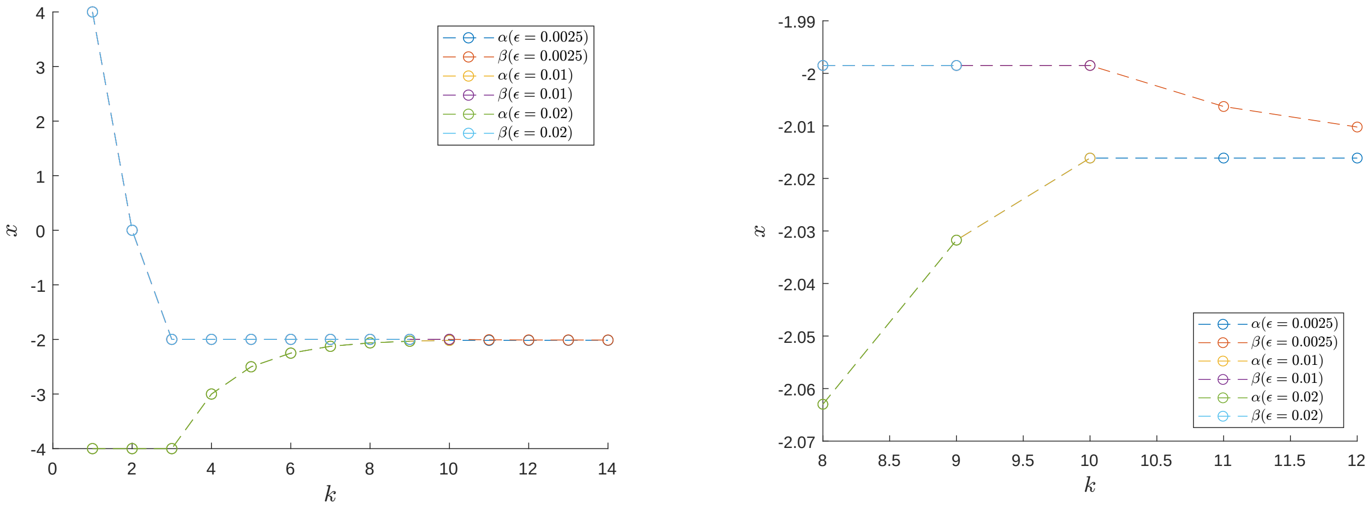
(b)

Εικόνα 3



(a) (b)

Εικόνα 4



(a) (b)

Εικόνα 5

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.

Θέμα 2 - Μέθοδος του Χρυσού Τομέα

Θέμα 3 - Μέθοδος Fibonacci

Θέμα 4 - Μέθοδος της Διχοτόμου με χρήση παραγώγου

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
fplot(f1(x),intervalStart);
fplot(f2(x),intervalStart);
fplot(f3(x),intervalStart);
legend

%% 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 = [0.0001 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 = [0.0001 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] = ...
            fibonacci(intervalStart,intervalStep,intervalAccuracy(i),funcs{idFunc});
        ends(idFunc,i) = {endPoints};
        countFunctionCalls(idFunc,i) = {count};
    end
end

%% BISECTION DERIVATIVE METHOD
intervalAccuracy = [0.0001 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
plot(intervalStep,cell2mat(countFunctionCalls(1,:)),'--o')
xlabel('$\epsilon$','fontsize',14,'interpreter','latex')
ylabel('Function calls $f_1(x)$','fontsize',14,'interpreter','latex')

figure
plot(intervalStep,cell2mat(countFunctionCalls(2,:)),'--o')
xlabel('$\epsilon$','fontsize',14,'interpreter','latex')
ylabel('Function calls $f_2(x)$','fontsize',14,'interpreter','latex')
```

```
figure
plot(intervalStep,cell2mat(countFunctionCalls(3,:)),'--o')
xlabel('$\epsilon$','fontsize',14,'interpreter','latex')
ylabel('Function calls $f_3(x)$','fontsize',14,'interpreter','latex')

%% 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('$\epsilon$','fontsize',14,'interpreter','latex')
ylabel('Function calls $f_1(x)$','fontsize',14,'interpreter','latex')
end

%% varying lambda
figure
plot(intervalAccuracy,cell2mat(countFunctionCalls(1,:)),'--o')
xlabel('$\lambda$','fontsize',14,'interpreter','latex')
ylabel('Function calls $f_1(x)$','fontsize',14,'interpreter','latex')

figure
plot(intervalAccuracy,cell2mat(countFunctionCalls(2,:)),'--o')
xlabel('$\lambda$','fontsize',14,'interpreter','latex')
ylabel('Function calls $f_2(x)$','fontsize',14,'interpreter','latex')

figure
plot(intervalAccuracy,cell2mat(countFunctionCalls(3,:)),'--o')
xlabel('$\lambda$','fontsize',14,'interpreter','latex')
ylabel('Function calls $f_3(x)$','fontsize',14,'interpreter','latex')

%% varying end points with stable epsilon
figure
hold on
intervalAccuracyIndex = 5;
for i = 1:3
pairs = cell2mat(ends(i,intervalAccuracyIndex));
[k,] = size(pairs);
plot(1:k,pairs(:,1),'--o')
plot(1:k,pairs(:,2),'--o')
xlabel('$\epsilon$','fontsize',14,'interpreter','latex')
ylabel('Function calls $f_1(x)$','fontsize',14,'interpreter','latex')
end

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

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

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

    % find number of iterations
    fib(1) = 0; fib(2) = 1;
    n = 2;
    while((fib(n)+fib(n-1)) ≤ (b(1)-a(1))/intervalAccuracy)
       n = n + 1;
       fib(n) = fib(n-2) + fib(n-1);
    end
    %fprintf("num iterations: %d, %0.2f\n",n,(b(1)-a(1))/intervalAccuracy)
```

```
    nextX = @(a,b,ratio) a + ratio*(b-a);
    x1 = nextX(a(1),b(1),fib(n-2)/fib(n));
    x2 = nextX(a(1),b(1),fib(n-1)/fib(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),fib(n-2-i)/fib(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),fib(n-1-i)/fib(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+1) > 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-1
        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
```