

# 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

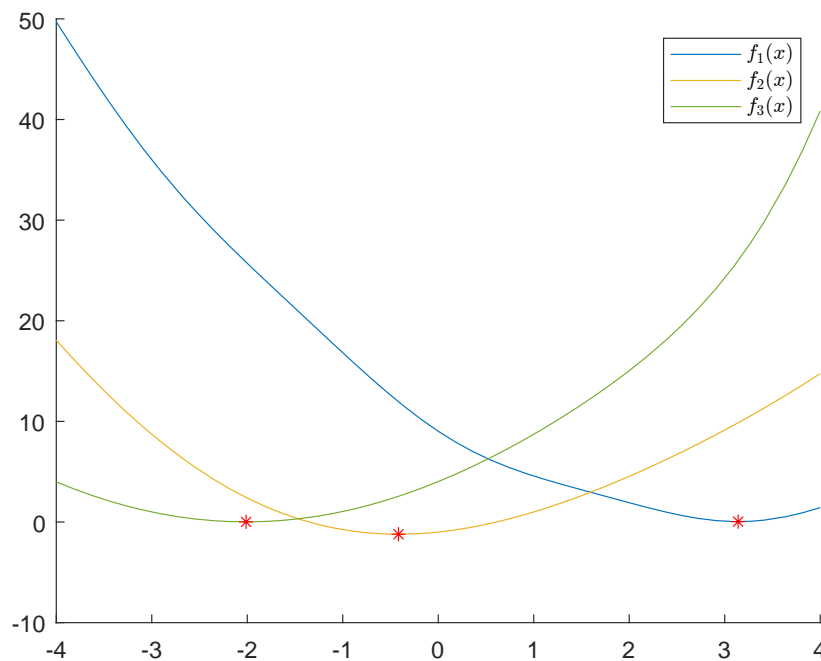
Σκοπός αυτής της εργασίας είναι η μελέτη διάφορων μεθόδων ελαχιστοποίησης κυρτής συνάρτησης  $f(x)$  σε δοσμένο διάστημα  $[\alpha, \beta]$ . Οι συναρτήσεις που θα μελετηθούν με αρχικό διάστημα  $[-4, 4]$  είναι οι εξής:

$$f_1(x) = (x - 3)^2 + \sin^2(x + 3) \quad (1)$$

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

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

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



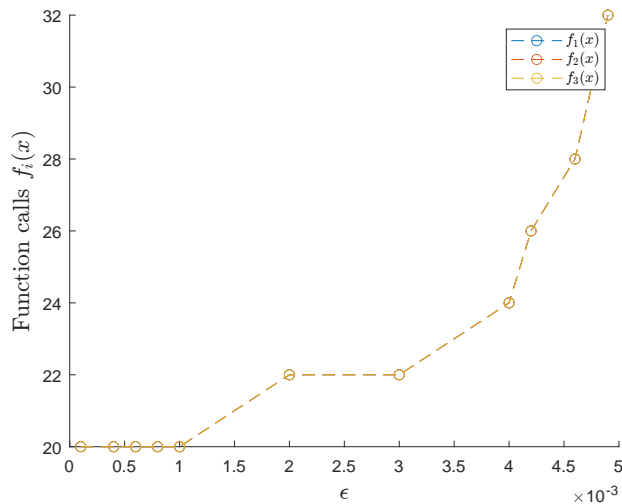
Εικόνα 1: Γραφική αναπαράσταση συναρτήσεων και των ελάχιστων σημείων τους

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  $x_1$  και  $x_2$ , αν  $f(x_1) < f(x_2)$  τότε το νέο διάστημα αναζήτησης θα είναι  $[\alpha, x_2]$  αλλιώς  $[x_1, \beta]$ . Με ανάλογο τρόπο, επαναλαμβάνουμε το προηγούμενο σκεπτικό μέχρι να φτάσουμε στο επιθυμητό διάστημα που υποδηλώνει την ακρίβεια της προσέγγισης μας.

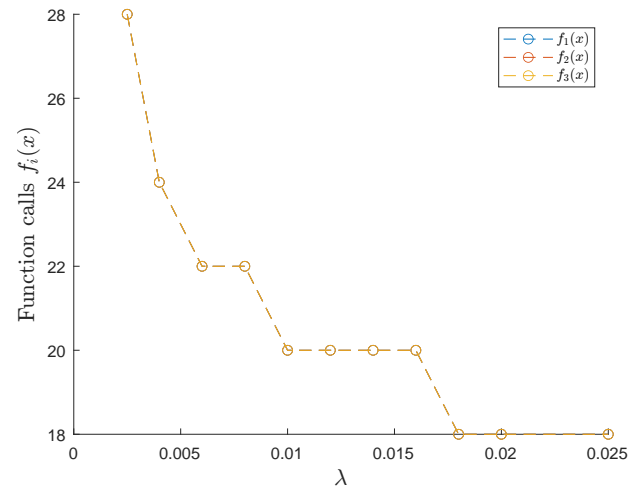
## Θέμα 1 - Μέθοδος της Διχοτόμου

Για την μέθοδο της διχοτόμου, έχοντας μια αυστηρή κυρτή συνάρτηση  $f(x)$ , όπου  $x \in [a, b]$ , επιλέγουμε δύο σημεία  $x_1$  και  $x_2$  εντός του διαστήματος, όπου  $x_1 = \frac{\beta+\alpha}{2} - \epsilon$  και  $x_2 = \frac{\beta+\alpha}{2} + \epsilon$  με  $\epsilon > 0$  θετική σταθερά. Η επιλογή αυτών των σημείων έγινε με γνώμονα την δημιουργία ίσου εύρους πιθανών διαστημάτων. Αξίζει να σημειωθεί ότι για  $\epsilon$ , η τελική ακρίβεια του διαστήματος ( $l$ ) δεν μπορεί να είναι μικρότερη από  $2\epsilon$  ( $l > 2\epsilon$ ). 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  $\epsilon$  και σταθερό  $l = 0.01$



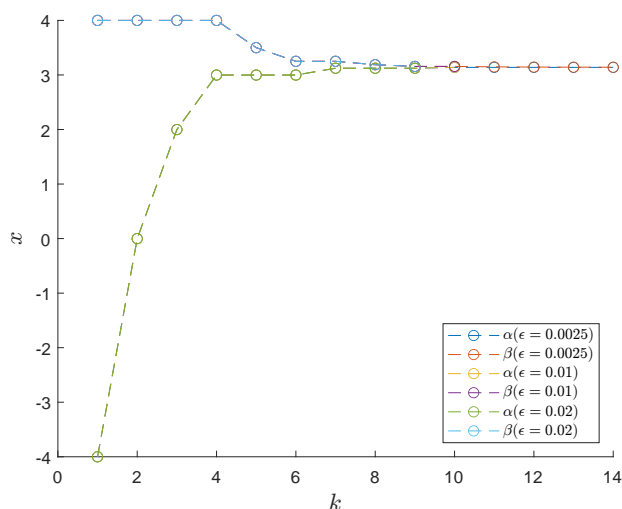
(b) Μεταβολή του αριθμού κλήσεων της συνάρτησης  $f$  μεταβάλλοντας την σταθερά  $l$  και σταθερό  $\epsilon = 0.001$

Εικόνα 2

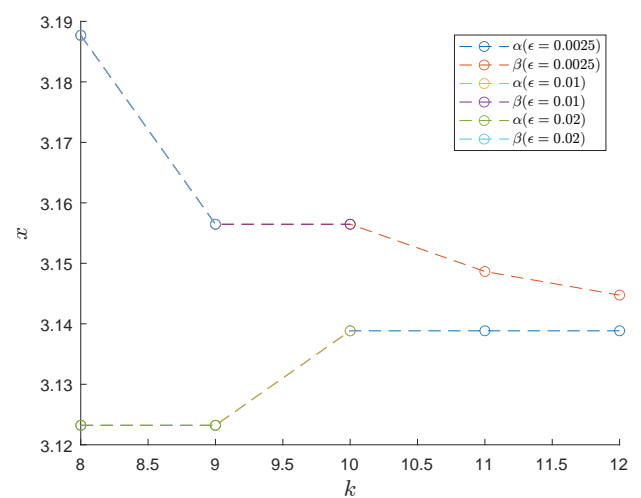
Παρατηρώντας την εικόνα 2, ο αριθμός των κλήσεων της αντικειμενικής συνάρτησης  $f_i(x)$  αυξάνεται όταν μειώνουμε την σταθερά  $\epsilon$  και όταν αυξάνουμε την σταθερά  $l$ . Αυτό φυσικά είναι αναμενόμενο καθώς μικρότερο  $\epsilon$  και ως συνεπακόλουθο μεγαλύτερο  $l$  σημαίνει μεγαλύτερη ακρίβεια, 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.

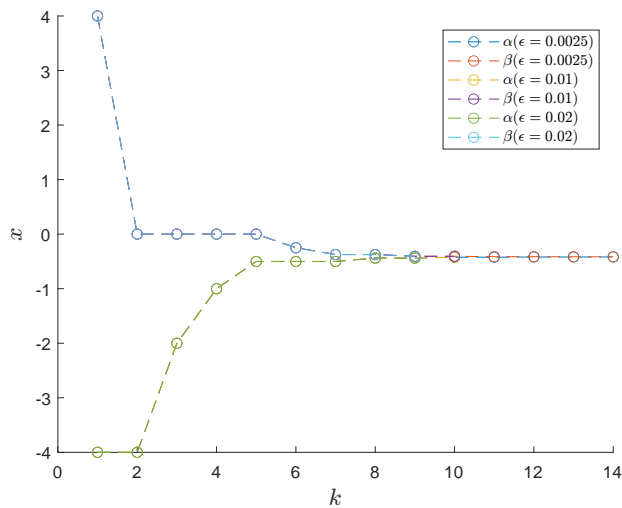


(a)

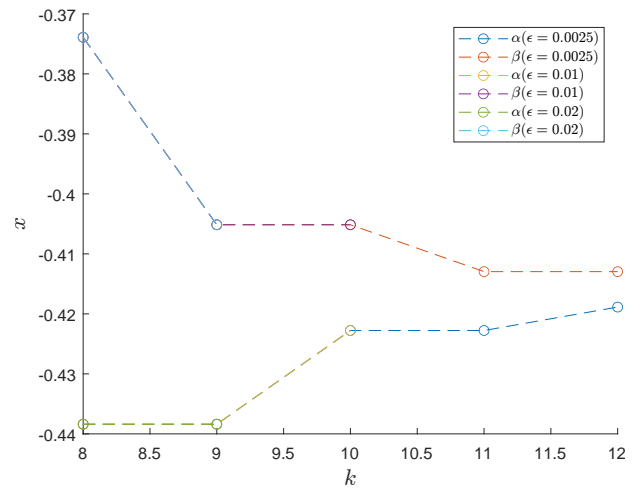


(b)

Εικόνα 3

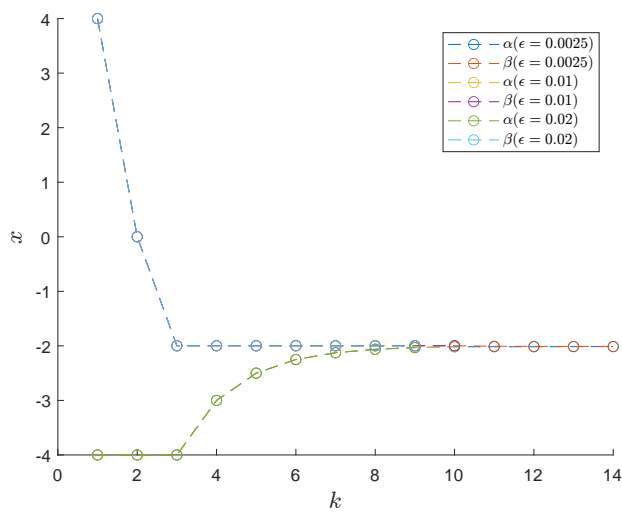


(a)

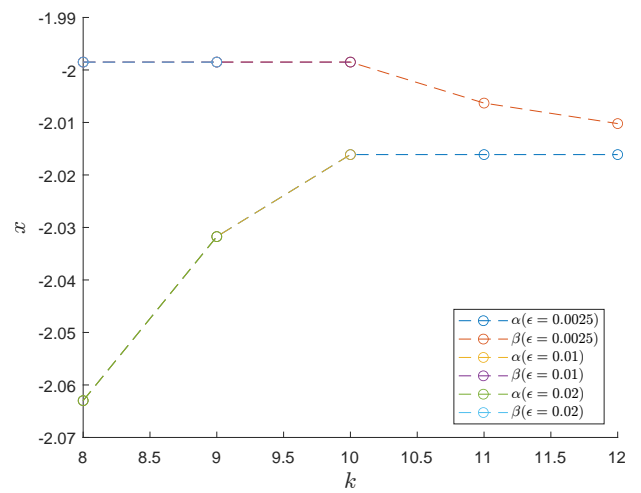


(b)

Εικόνα 4



(a)



(b)

Εικόνα 5

Όπως είναι αναμενόμενο, τα άκρα  $\alpha, \beta$  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  $\alpha$  και  $\beta$ . Πιο συγκεκριμένα, βλέπουμε συνολικά ότι για κάθε συνάρτηση, τα άκρα  $\alpha, \beta$  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.

```
1 %% MINIMIZATION PROBLEM FOR CONVEX FUNCTIONS SINGLE VARIABLE
2 clc
3 clear all
4
5 % FUNCTIONS
6 f1 = @(x) (x-3)^2 + (sin(x+3))^2;
7 f2 = @(x) (x-1) * cos(0.5 * x) + x^2;
8 f3 = @(x) (x+2)^2 + exp(x-2) * sin(x+3);
9 numFuncs = 3;
10
11 intervalStart = [-4 4];
12
13 syms x;
14 figure
15 hold on
16 fplot(f1(x),intervalStart);
17 fplot(f2(x),intervalStart);
18 fplot(f3(x),intervalStart);
19 legend
20
21 %% BISECTION METHOD
22 % STABLE ACCURACY (lambda) VARIABLE STEP (epsilon)
23 intervalAccuracy = 0.01;
24 intervalStep = [0.0001 0.0004 0.0006 0.0008 0.001 0.002 0.003 0.004 0.0042 ...
25 0.0046 0.0049];
26 ends = cell(numFuncs,numel(intervalStep));
27 countFunctionCalls = cell(numFuncs,numel(intervalStep));
28
29 funcs=cell(numFuncs,1);
30 funcs{1} = f1; funcs{2} = f2; funcs{3} = f3;
31 for idFunc=1:numel(funcs)
32     for i=1:numel(intervalStep)
33         [endPoints,count] = ...
34             bisection(intervalStart,intervalStep(i),intervalAccuracy,funcs{idFunc});
35         ends(idFunc,i) = {endPoints};
36         countFunctionCalls(idFunc,i) = {count};
37     end
38 end
39
40 %% BISECTION METHOD
41 % STABLE STEP (epsilon) VARIABLE ACCURACY (lambda)
42 intervalStep = 0.001;
43 intervalAccuracy = [0.0025 0.004 0.006 0.008 0.01 0.012 0.014 0.016 0.018 0.02 ...
44 0.025];
45 ends = cell(numFuncs,numel(intervalAccuracy));
46 countFunctionCalls = cell(numFuncs,numel(intervalAccuracy));
47
48 funcs=cell(numFuncs,1);
49 funcs{1} = f1; funcs{2} = f2; funcs{3} = f3;
50 for idFunc=1:numel(funcs)
51     for i=1:numel(intervalAccuracy)
52         [endPoints,count] = ...
53             bisection(intervalStart,intervalStep,intervalAccuracy(i),funcs{idFunc});
54         ends(idFunc,i) = {endPoints};
55         countFunctionCalls(idFunc,i) = {count};
56     end
57 end
```

```

54
55 %% GOLDEN SECTOR METHOD
56 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];
57 ends = cell(numFuncs,numel(intervalAccuracy));
58 countFunctionCalls = cell(numFuncs,numel(intervalAccuracy));
59
60 funcs=cell(numFuncs,1);
61 funcs{1} = f1; funcs{2} = f2; funcs{3} = f3;
62 for idFunc=1:numel(funcs)
63     for i=1:numel(intervalAccuracy)
64         [endPoints,count] = ...
            goldenSector(intervalStart,intervalAccuracy(i),funcs{idFunc});
65         ends(idFunc,i) = {endPoints};
66         countFunctionCalls(idFunc,i) = {count};
67     end
68 end
69
70 %% FIBONACCI METHOD
71 intervalStep = 0.001;
72 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];
73 ends = cell(numFuncs,numel(intervalAccuracy));
74 countFunctionCalls = cell(3,numel(intervalAccuracy));
75
76 funcs=cell(numFuncs,1);
77 funcs{1} = f1; funcs{2} = f2; funcs{3} = f3;
78 for idFunc=1:numel(funcs)
79     for i=1:numel(intervalAccuracy)
80         [endPoints,count] = ...
            fibonacci(intervalStart,intervalStep,intervalAccuracy(i),funcs{idFunc});
81         ends(idFunc,i) = {endPoints};
82         countFunctionCalls(idFunc,i) = {count};
83     end
84 end
85
86 %% BISECTION DERIVATIVE METHOD
87 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];
88 ends = cell(numFuncs,numel(intervalAccuracy));
89 countFunctionCalls = cell(numFuncs,numel(intervalAccuracy));
90
91 funcs=cell(numFuncs,1);
92 funcs{1} = f1; funcs{2} = f2; funcs{3} = f3;
93 for idFunc=1:numel(funcs)
94     for i=1:numel(intervalAccuracy)
95         [endPoints,count] = ...
            bisectionDerivative(intervalStart,intervalAccuracy(i),funcs{idFunc});
96         ends(idFunc,i) = {endPoints};
97         countFunctionCalls(idFunc,i) = {count};
98     end
99 end
100 %% PLOTS
101 % varying epsilon (used for bisection method)
102 figure
103 plot(intervalStep,cell2mat(countFunctionCalls(1,:)),'—o')
104 xlabel('$\epsilon$', 'fontsize',14, 'interpreter','latex')
105 ylabel('Function calls $f_1(x)$', 'fontsize',14, 'interpreter','latex')
106
107 figure
108 plot(intervalStep,cell2mat(countFunctionCalls(2,:)),'—o')
109 xlabel('$\epsilon$', 'fontsize',14, 'interpreter','latex')
110 ylabel('Function calls $f_2(x)$', 'fontsize',14, 'interpreter','latex')

```

```

111 figure
112 plot(intervalStep, cell2mat(countFunctionCalls(3,:)), '—o')
113 xlabel('$\epsilon$', 'fontsize', 14, 'interpreter', 'latex')
114 ylabel('Function calls $f_3(x)$', 'fontsize', 14, 'interpreter', 'latex')
115
116 %% varying end points with stable lambda
117 figure
118 hold on
119 intervalStepIndex = 11;
120 for i = 1:3
121     pairs = cell2mat(ends(i, intervalStepIndex));
122     [k,] = size(pairs);
123     plot(1:k, pairs(:,1), '—o')
124     plot(1:k, pairs(:,2), '—o')
125     xlabel('$\epsilon$', 'fontsize', 14, 'interpreter', 'latex')
126     ylabel('Function calls $f_1(x)$', 'fontsize', 14, 'interpreter', 'latex')
127 end
128
129 %% varying lambda
130 figure
131 plot(intervalAccuracy, cell2mat(countFunctionCalls(1,:)), '—o')
132 xlabel('$\lambda$', 'fontsize', 14, 'interpreter', 'latex')
133 ylabel('Function calls $f_1(x)$', 'fontsize', 14, 'interpreter', 'latex')
134
135 figure
136 plot(intervalAccuracy, cell2mat(countFunctionCalls(2,:)), '—o')
137 xlabel('$\lambda$', 'fontsize', 14, 'interpreter', 'latex')
138 ylabel('Function calls $f_2(x)$', 'fontsize', 14, 'interpreter', 'latex')
139
140 figure
141 plot(intervalAccuracy, cell2mat(countFunctionCalls(3,:)), '—o')
142 xlabel('$\lambda$', 'fontsize', 14, 'interpreter', 'latex')
143 ylabel('Function calls $f_3(x)$', 'fontsize', 14, 'interpreter', 'latex')
144
145 %% varying end points with stable epsilon
146 figure
147 hold on
148 intervalAccuracyIndex = 5;
149 for i = 1:3
150     pairs = cell2mat(ends(i, intervalAccuracyIndex));
151     [k,] = size(pairs);
152     plot(1:k, pairs(:,1), '—o')
153     plot(1:k, pairs(:,2), '—o')
154     xlabel('$\epsilon$', 'fontsize', 14, 'interpreter', 'latex')
155     ylabel('Function calls $f_1(x)$', 'fontsize', 14, 'interpreter', 'latex')
156 end
157
158 function [ends, countFunctionCalls] = ...
159     bisection(intervalStart, intervalStep, intervalAccuracy, f)
160     a(1) = intervalStart(1);
161     b(1) = intervalStart(2);
162     countFunctionCalls = 0;
163     i = 1;
164     while b(i) - a(i) >= intervalAccuracy
165         midPoint = (a(i) + b(i)) / 2;
166         x1 = midPoint - intervalStep;
167         x2 = midPoint + intervalStep;
168         if f(x1) < f(x2)
169             a(i+1) = a(i);
170             b(i+1) = x2;
171         else
172             a(i+1) = x1;

```

```

173         b(i+1) = b(i);
174     end
175     countFunctionCalls = countFunctionCalls+2;
176     i = i+1;
177 end
178
179 ends(:,1) = a(1:end-1);
180 ends(:,2) = b(1:end-1);
181 end
182
183 function [ends, countFunctionCalls] = ...
    goldenSector(intervalStart, intervalAccuracy, f)
184 % init
185 gamma = 0.618;
186 a(1) = intervalStart(1);
187 b(1) = intervalStart(2);
188 nextX1 = @(a,b,gamma) a + (1-gamma)*(b-a);
189 nextX2 = @(a,b,gamma) a + gamma*(b-a);
190 x1 = nextX1(a(1), b(1), gamma);
191 x2 = nextX2(a(1), b(1), gamma);
192 y1 = f(x1);
193 y2 = f(x2);
194
195 i = 1;
196 countFunctionCalls = 2;
197 while (b(i)-a(i)) >= intervalAccuracy
198     if y1 < y2
199         a(i+1) = a(i);
200         b(i+1) = x2;
201         x2 = x1;
202         x1 = nextX1(a(i+1), b(i+1), gamma);
203         y2 = y1;
204         y1 = f(x1);
205     else
206         a(i+1) = x1;
207         b(i+1) = b(i);
208         x1 = x2;
209         x2 = nextX2(a(i+1), b(i+1), gamma);
210         y1 = y2;
211         y2 = f(x2);
212     end
213     countFunctionCalls = countFunctionCalls+1;
214     i = i+1;
215 end
216
217 ends(:,1) = a(1:end-1);
218 ends(:,2) = b(1:end-1);
219 end
220
221 function [ends, countFunctionCalls] = ...
    fibonacci(intervalStart, intervalStep, intervalAccuracy, f)
222 % init
223 a(1) = intervalStart(1);
224 b(1) = intervalStart(2);
225
226 % find number of iterations
227 fib(1) = 0; fib(2) = 1;
228 n = 2;
229 while ((fib(n)+fib(n-1)) <= (b(1)-a(1))/intervalAccuracy)
230     n = n + 1;
231     fib(n) = fib(n-2) + fib(n-1);
232 end
233 %fprintf("num iterations: %d, %0.2f\n", n, (b(1)-a(1))/intervalAccuracy)

```



```

234
235     nextX = @(a,b, ratio) a + ratio*(b-a);
236     x1 = nextX(a(1),b(1),fib(n-2)/fib(n));
237     x2 = nextX(a(1),b(1),fib(n-1)/fib(n));
238     y1 = f(x1);
239     y2 = f(x2);
240
241     i = 1;
242     countFunctionCalls = 2;
243     while i ≤ n-3
244         if y1 < y2
245             if i == (n-3)
246                 a(i+1) = a(i);
247                 b(i+1) = x1;
248             else
249                 a(i+1) = a(i);
250                 b(i+1) = x2;
251                 x2 = x1;
252                 x1 = nextX(a(i+1),b(i+1),fib(n-2-i)/fib(n-i));
253                 y2 = y1;
254                 y1 = f(x1);
255             end
256         else
257             if i == (n-3)
258                 a(i+1) = x1;
259                 b(i+1) = b(i);
260             else
261                 a(i+1) = x1;
262                 b(i+1) = b(i);
263                 x1 = x2;
264                 x2 = nextX(a(i+1),b(i+1),fib(n-1-i)/fib(n-i));
265                 y1 = y2;
266                 y2 = f(x2);
267             end
268         end
269         countFunctionCalls = countFunctionCalls+1;
270         i = i+1;
271         if i == (n-3)
272             x2 = x1 + intervalStep;
273             y2 = f(x2);
274         end
275     end
276
277     ends(:,1) = a;
278     ends(:,2) = b;
279 end
280
281 function [ends, countFunctionCalls] = ...
    bisectionDerivative(intervalStart, intervalAccuracy, f)
282     a(1) = intervalStart(1);
283     b(1) = intervalStart(2);
284
285     % derivative
286     df = matlabFunction(diff(sym(f)));
287
288     % find number of iterations
289     n = 1;
290     while ((0.5)^(n+1) > intervalAccuracy/(b(1)-a(1)))
291         n = n+1;
292     end
293     %fprintf("%0.2f,%d\n", intervalAccuracy/(b(1)-a(1)), n)
294
295     countFunctionCalls = 0;

```

```

296     i = 1;
297     while i ≤ n-1
298         x = (a(i)+b(i))/2;
299         dx = df(x);
300         countFunctionCalls = countFunctionCalls+1;
301         if dx == 0
302             a(i+1) = x;
303             b(i+1) = x;
304             break
305         elseif dx < 0
306             a(i+1) = x;
307             b(i+1) = b(i);
308         elseif dx > 0
309             a(i+1) = a(i);
310             b(i+1) = x;
311         end
312         i = i+1;
313     end
314
315     ends(:,1) = a;
316     ends(:,2) = b;
317 end

```