Τεχνικές Βελτιστοποίησης — 1ο Παραδοτέο

Αναζήτηση ελαχίστου: Διχοτόμος, Χρυσή Τομή, Fibonacci, Διχοτόμος με Παράγωγο

Χρήστος Αλεξόπουλος-10618

Contents

1 Εισαγωγή 2

2 Μέθοδος Διχοτόμου 2

3 Μέθοδος Χρυσής Τομής 4

4 Μέθοδος Fibonacci 5

5 Διχοτόμος με χρήση παραγώγου 7

Παρατηρήσεις &Συμπεράσματα 10

Συγκριτική Αξιολόγηση Μεθόδων 12

Αρχιτεκτονική &Οργάνωση Κώδικα 12

1 Εισαγωγή

Στόχος της εργασίας είναι η πειραματική μελέτη τεσσάρων κλασικών μεθόδων μονοδιάστατης αναζήτησης ελαχίστου στο $[a, b]$: Διχοτόμος, Χρυσή Τομή, Fibonacci, Διχοτόμος με παράγωγο. Μετράμε τον αριθμό κλήσεων της συνάρτησηςf ως προς l(το μήκος του διαστήματος ενδιαφέροντος) και ως προς ε(ανεκτικότητα) και προβάλλουμε την αλλαγή του διαστήματος (a_k, b_k) για κάθε επανάληψη της μεθόδου. Η υλοποίηση των μεθόδων βασίζεται στις παραγράφους του βιβλίου 5.1.1, 5.1.2, 5.1.3, 5.1.4.

2 Μέθοδος Διχοτόμου

1) Σύνοψη

Σε κάθε βήμα: $m = \frac{a_k+b_k}{2}$, $x_{1k} = m - \varepsilon$, $x_{2k} = m + \varepsilon$. Αν $f(x_{1k}) < f(x_{2k})$ κρατάμε $[a_k, x_{2k}]$, αλλιώς $[x_{1k}, b_k]$. Μήκος: $L_{k+1} = \frac{L_k}{2} + \varepsilon \Rightarrow L_k = (L_0 - 2\varepsilon)2^{-k} + 2\varepsilon$. Τερματισμός όταν $L_k \leq l$:

$$k = \left\lceil \log_2 \frac{L_0 - 2\varepsilon}{l - 2\varepsilon} \right\rceil$$

, $\#f\text{-calls} = 2k$ (απαιτείται $l > 2\varepsilon$).

2) Αποτελέσματα

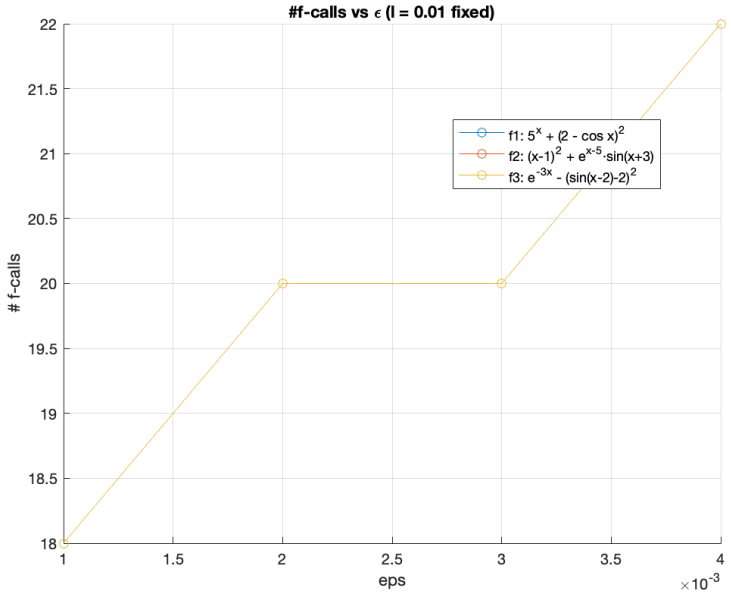


Figure 1: $\#f$-calls vs ε (σταθερό l).

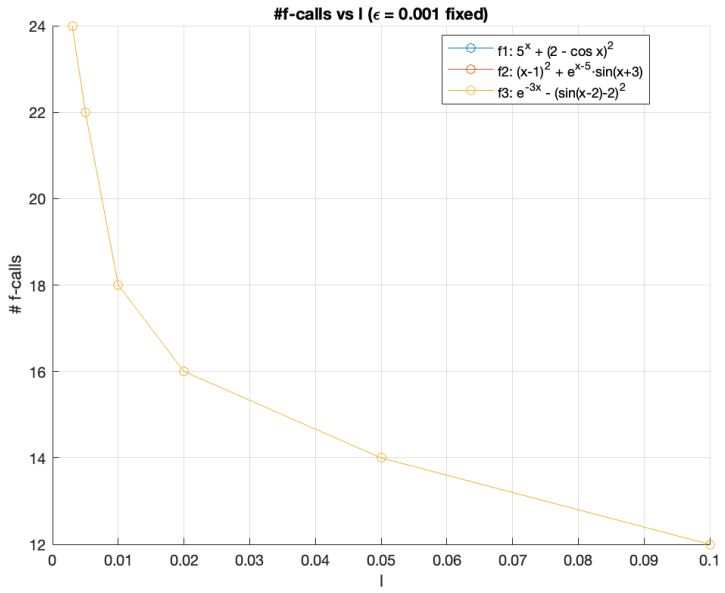
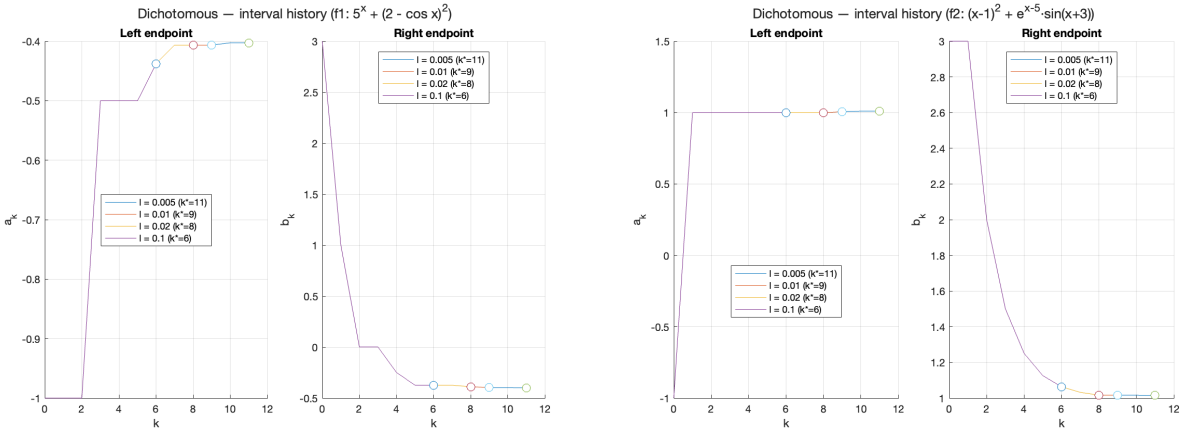
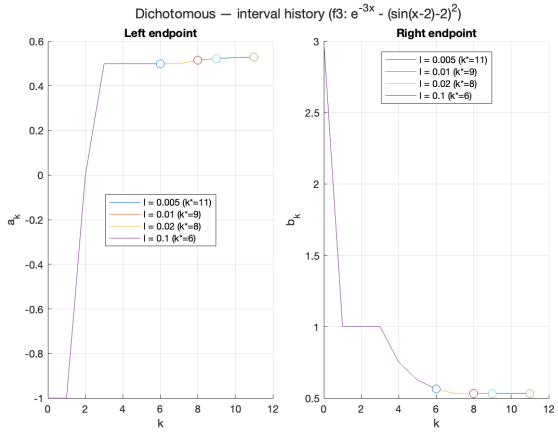


Figure 2: $\#f$-calls vs ε (σταθερό ε).



(a) f_1

(b) f_2



(c) f_3

Figure 3: Αλλαγή $[a_k, b_k]$ για πολλές τιμές l (σταθερό ε).

3 Μέθοδος Χρυσής Τομής

1) Σύνοψη

Ορίζουμε $\gamma = \frac{\sqrt{5}-1}{2} \approx 0.618$. Αρχικοποίηση: $x_{1,1} = a_1 + (1-\gamma)(b_1 - a_1)$, $x_{2,1} = a_1 + \gamma(b_1 - a_1)$. Reuse μίας τιμής 1 νέα κλήση/βήμα. Σύσπαση: $L_{k+1} = \gamma L_k \Rightarrow L_k = \gamma^k L_0$. Τερματισμός όταν $L_k \leq l$:

$$k = \left\lceil \frac{\ln(l/L_0)}{\ln \gamma} \right\rceil$$

, $\#f\text{-calls} = 2 + k$.

2) Αποτελέσματα

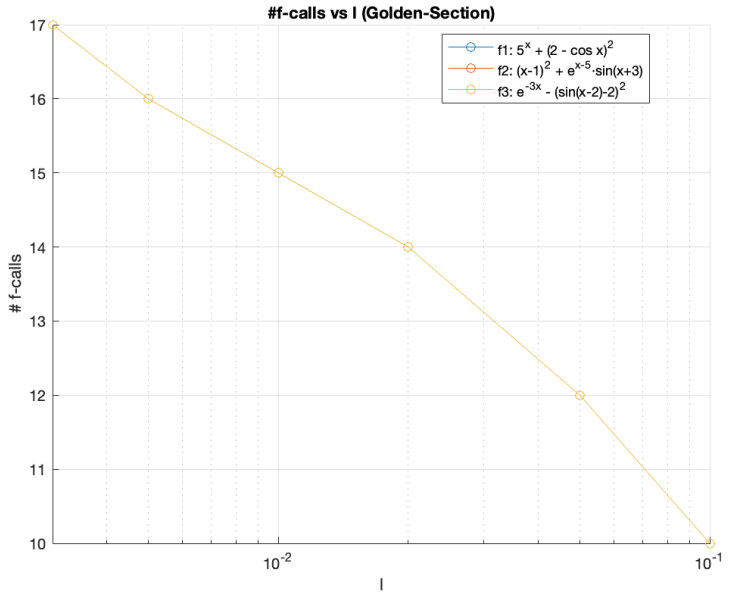
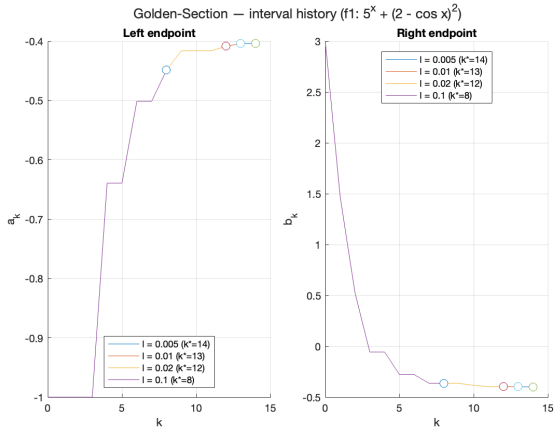
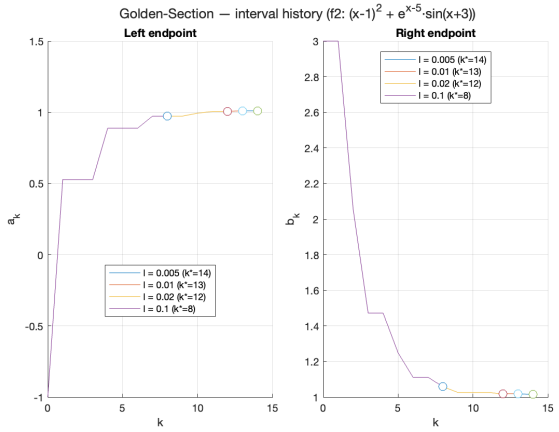


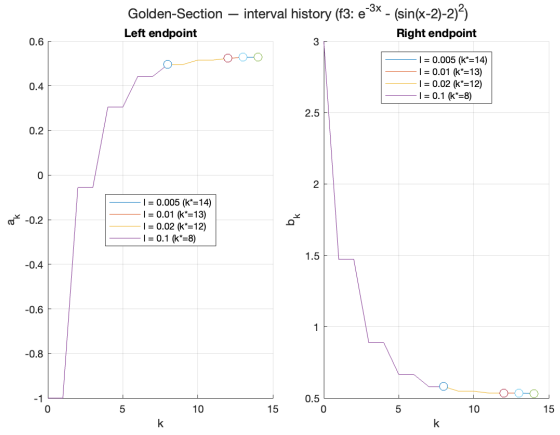
Figure 4: $\#f$-calls vs l (Χρυσή Τομή).



(a) f_1



(b) f_2



(c) f_3

Figure 5: Αλλαγή $[a_k, b_k]$ για πολλές τιμές l (Χρυσή Τομή).

4 Μέθοδος Fibonacci

1) Σύνοψη

Διαλέγουμε ελάχιστο n με $F_n > L_0/l$. Αρχικοποίηση: $x_{1,1} = a_1 + \frac{F_{n-2}}{F_n}(b_1 - a_1)$, $x_{2,1} = a_1 + \frac{F_{n-1}}{F_n}(b_1 - a_1)$. Reuse 1 νέα κλήση/βήμα. Μήκος: $L_k = \frac{F_{n-k}}{F_n} L_0 \Rightarrow L_n \leq L_0/F_n \leq l$. Με σωστό reuse στο τέλος: $\#f\text{-calls} = n$.

2) Αποτελέσματα

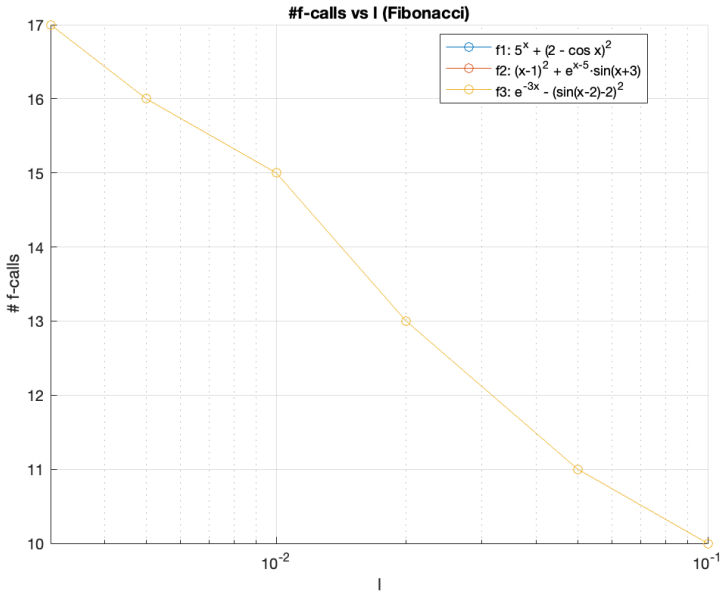
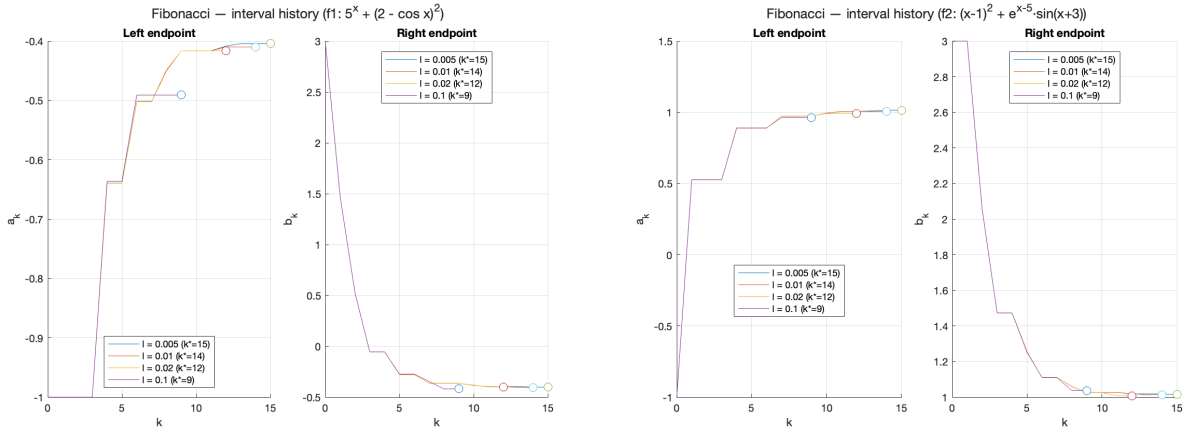
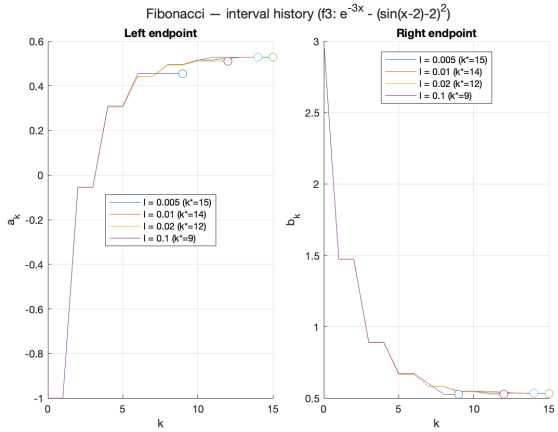


Figure 6: $\#f$-calls vs l (Fibonacci) — «σκαλοπάτια» λόγω ακέραιου n.



(a) f_1

(b) f_2



(c) f_3

Figure 7: Αλλαγή $[a_k, b_k]$ για πολλές τιμές l (Fibonacci).

5 Διχοτόμος με χρήση παραγώγου

1) Σύνοψη

Σε $x_k = \frac{a_k+b_k}{2}$: $f'(x_k) > 0 \Rightarrow [a_{k+1}, b_{k+1}] = [a_k, x_k]$, $f'(x_k) < 0 \Rightarrow [a_{k+1}, b_{k+1}] = [x_k, b_k]$. Μήκος: $L_k = L_0/2^k$. Με αριθμητική παράγωγο (κεντρικό διαφορικό) 2 κλήσεις f/βήμα: $\#f\text{-calls} \approx 2\lceil \log_2(L_0/l) \rceil$.

2) Αποτελέσματα

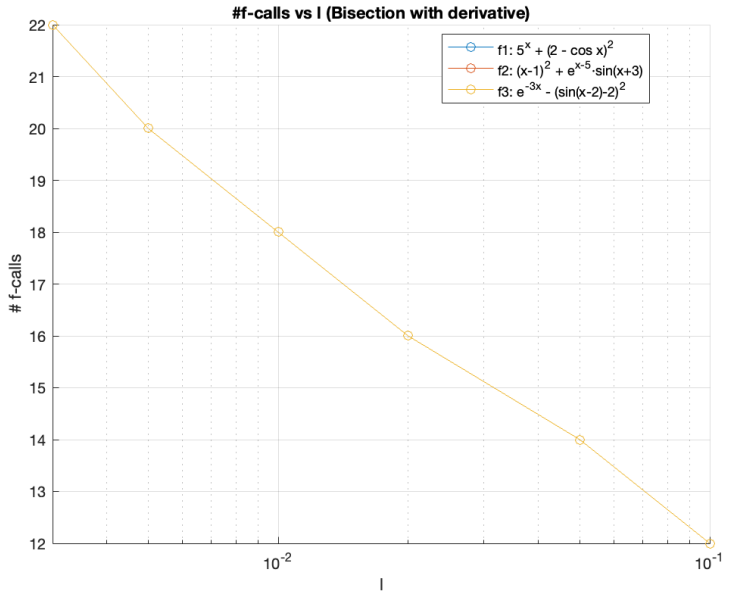
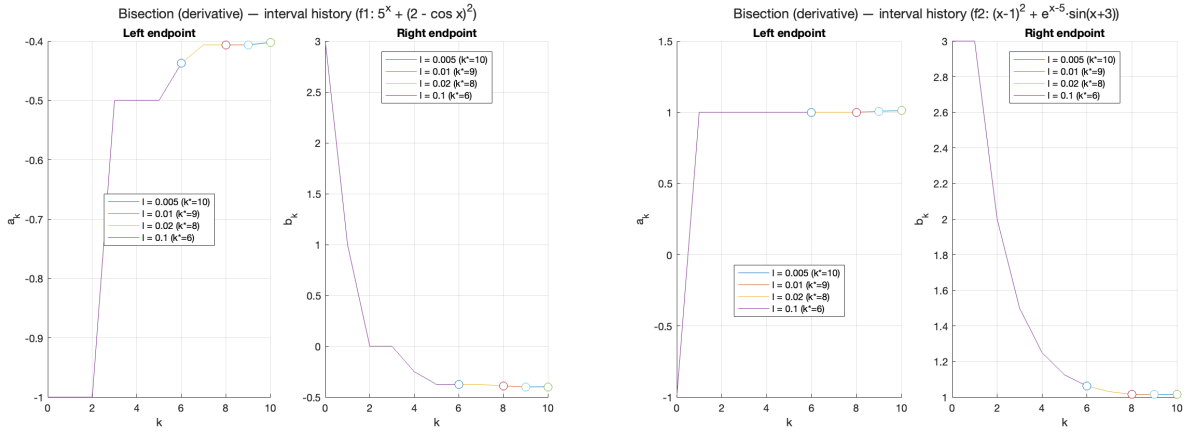
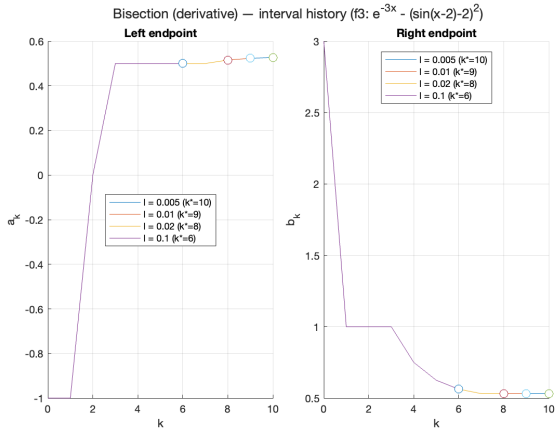


Figure 8: $\#f$-calls vs l (αριθμητική παράγωγος).



(a) f_1

(b) f_2



(c) f_3

Figure 9: Αλλαγή $[a_k, b_k]$ για πολλές τιμές l (Διχοτόμος με παράγωγο).

Παρατηρήσεις & Συμπεράσματα

1) Σε όλες τις μεθόδους παρατηρούμε ότι τα $\#f$-calls **μειώνονται** καθώς το l **μεγαλώνει**. Αυτό είναι αναμενόμενο: όσο πιο «χαλαρή» είναι η τελική ακρίβεια διαστήματος, τόσο λιγότερα βήματα χρειαζόμαστε για να ολοκληρωθεί η μέθοδος αφού η συνθήκη για το αν είμαστε μέσα στο τελικό διάστημα συναντάται πιο γρήγορα.

2) Σε όλες τις μεθόδους βλέπουμε μία καμπύλη αντί για 3 στα διαγράμματα των #f-calls. Τα $\#f$-calls ως προς l εξαρτώνται μόνο από **γεωμετρία σύσπασης** (*ο κανόνας με τον οποίο μικραίνει συστηματικά το μήκος του διαστήματος, π.χ.* $L_{k+1} = rL_k$ *ή* $L_{k+1} = \frac{1}{2}L_k + \varepsilon$) και το **σταθερό κόστος ανά βήμα** (*πόσες αξιολογήσεις της* f *γίνονται σε κάθε επανάληψη μετά την αρχικοποίηση*), άρα οι καμπύλες για f_1, f_2, f_3 επικαλύπτονται.

Οπτική επαλήθευση. Οι τρεις συναρτήσεις που χρησιμοποιήθηκαν έχουν παρόμοια μορφή στο $[-1, 3]$.

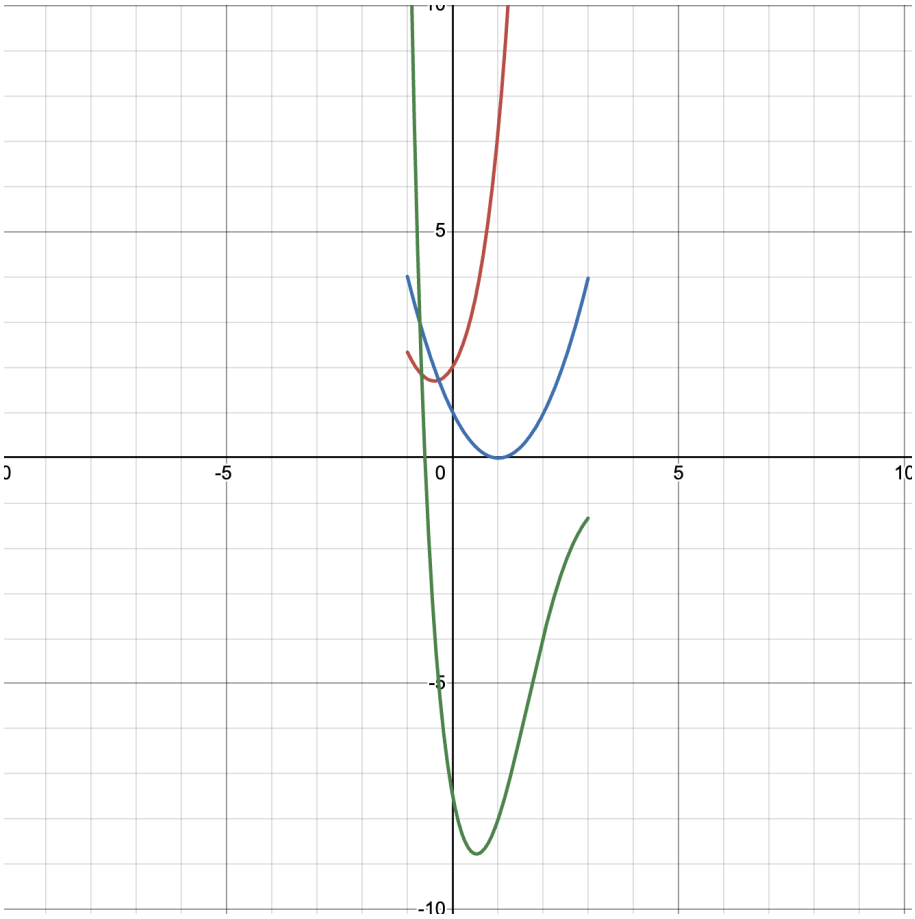


Figure 10: Οπτικοποίηση των f_1, f_2, f_3 (Desmos). Οι καμπύλες είναι μονοκορυφικές στο $[-1, 3]$, οπότε οι μέθοδοι σύσπασης εμφανίζουν ίδια καμπύλη $\#f$-calls ως προς l.

3) Για τα διαστήματα $[a_k, b_k]$ ως συνάρτηση του k βλέπουμε ότι, για κάθε μέθοδο και για σταθερή f_i, οι καμπύλες για διαφορετικά l δίνουν την ίδια γραφική παράσταση και απλώς τερματίζουν σε διαφορετικό k (άρα και σε διαφορετικά τελικά άκρα). Με απλά λόγια για μικρότερο l απαιτείται μεγαλύτερο $k^{(l)}$ με ίδια τροχιά $[a_k, b_k]$ αλλά περισσότερα βήματα πριν να τελειώσουμε την αναζήτηση.

4) Επαληθεύουμε ότι οι μέθοδοι εντοπίζουν τα ελάχιστα σημεία τα οποία βλέπουμε από τα διαγράμματα των $[a_k, b_k]$ και είναι και αυτά που βλέπουμε και στο Desmos, τα εκτιμώμενα ελάχιστα είναι:

$$x^{f_1} \approx -0.4014, \qquad x^{f_2} \approx 1.01307, \qquad x^{f_3} \approx 0.53116.$$

Η σύγκλιση των $[a_k, b_k]$ στα σημεία φαίνεται στα αντίστοιχα plots.

Συγκριτική Αξιολόγηση Μεθόδων

Παρατηρούμε ότι για το πλήθος κλήσεων της f ισχύει η παρακάτω σχέση:

$$\textbf{Fibonacci} \lesssim \textbf{Χρυσή Τομή} < \textbf{Διχοτόμος με Παράγωγο} < \textbf{Διχοτόμος}.$$

Η Fibonacci και η Χρυσή τομή χρειάζονται τις λιγότερες κλήσεις, η Διχοτόμος με παράγωγο έχει την αμέσως καλύτερη απόδοση και η κλασική Διχοτόμος την χειρότερη απο τις 4.

Ποιοτική σύγκριση.

- **Fibonacci vs Χρυσή Τομή:** Και οι δύο κάνουν 1 νέα αξιολόγηση/βήμα. Η Fibonacci επιλέγει βέλτιστη σύσπαση κατά βήμα με βάση τους F_n.
- **Διχοτόμος με Παράγωγο:** Γραμμικός ρυθμός αλλά επειδή χρησιμοποιούμε κεντρική διαφορά για τον υπολογισμό της παραγώγου έχουμε 2 κλήσεις/βήμα. Αν χρησιμοποιούσαμε *αναλυτική* παράγωγο, οι κλήσεις της συνάρτησης f θα ήταν 0 ανα βήμα.
- **Διχοτόμος:** Δύο αξιολογήσεις/βήμα και σύσπαση με τον πρόσθετο όρο ε, άρα λιγότερο αποδοτική. Επίσης χρειάζεται να προσέξουμε να θέσουμε τον περιορισμό $l > 2\varepsilon$.

Συμπέρασμα:
Για ίδιες απαιτήσεις ακρίβειας l και για αυτές τις f που έχουμε, η **Fibonacci** και η **Χρυσή Τομή** είναι γενικά πιο αποδοτικές(λιγότερες φορές καλούμε την f). Η **Διχοτόμος με Παράγωγο** είναι καλύτερη όταν υπάρχει *αναλυτική* παράγωγος(θα μπορούσαμε να υπολογίσουμε τις αναλυτικές παραγώγους(αφου δίνονται αναλυτικά οι f) απλά το κάναμε με την κεντιρκή διαφορά για λόγους γενίκευσης και το σχολιάζουμε εδω). Η **Διχοτόμος** είναι η πιο απλή αλλά χρειάζεται τις περισσότερες κλήσεις.

Αρχιτεκτονική & Οργάνωση Κώδικα

Δομή φακέλων.

```
project/
  main.m
  figs/                          % όλα τα παραγόμενα διαγράμματα
  src/
    methods/                     % υλοποιήσεις αλγορίθμων
      dichotomous.m
      golden_section.m
      fibonacci.m
      bisection_derivative.m
    experiments/                 % scripts που τρέχουν πειράματα & αποθηκεύουν σχήματα
      ex_dichotomous.m
      ex_golden.m
      ex_fibonacci.m
```

```
  ex_bisection_derivative.m
utils/                        % κοινές βοηθητικές ρουτίνες
  count_calls_wrapper.m
  harness_run.m
  plot_interval_history.m
  l_vs_eps.m
test_functions.m              % ορισμός f1,f2,f3 και κοινού διαστήματος [a,b]
```

Ενιαίο interface μεθόδων. Κάθε μέθοδος υλοποιείται ως συνάρτηση:

$$[\, xmin,\ a\_hist,\ b\_hist,\ iters,\ f\_calls\,] = \texttt{method}(f, a1, b1, cfg)$$

όπου:

- `f` είναι function handle της αντικειμενικής συνάρτησης,
- `[a1,b1]` το αρχικό διάστημα,
- `cfg` περιέχει τις παραμέτρους της μεθόδου (π.χ. `l`, `eps`, `h`, `df`).
- Επιστρέφονται: εκτίμηση ελαχίστου (`xmin`), πορεία των άκρων (`a_hist`, `b_hist`), επαναλήψεις (`iters`), και `f_calls`.

Μετρητής κλήσεων f. Η `count_calls_wrapper.m` τυλίγει την `f` και αυξάνει μετρητή κάθε φορά που καλείται. Έτσι μετράμε ομοιόμορφα το κόστος ανεξάρτητα από τη μέθοδο.

Harness εκτέλεση. Το `harness_run.m`:

- τρέχει μια μέθοδο για όλες τις f_i και για όλες τις τιμές μιας παραμέτρου (π.χ. l, ε),
- καταγράφει `a_hist`, `b_hist`, `iters`, $\#f$-calls,
- επιστρέφει δομή `results` με `entries` ανά run (μας βολεύει για plotting).

Experiments & plots. Τα αρχεία στο `src/experiments/` ορίζουν *ποιες σαρώσεις* κάνουμε (π.χ. $\#f$-calls vs l, $\#f$-calls vs ε, "ιστορίες" $[a_k, b_k]$) και αποθηκεύουν σχήματα στον φάκελο `figs/`.

`plot_interval_history.m`: εμφανίζει $[a_k]$ και $[b_k]$ ανά k για πολλαπλά l στο ίδιο figure.

Συναρτήσεις δοκιμής. Το `test_functions.m` ορίζει τις $\{f_1, f_2, f_3\}$ και το κοινό αρχικό διάστημα $[-1, 3]$.

Σημείωση για το `cfg`

Το `cfg` είναι το «πακέτο ρυθμίσεων» που περνά σε κάθε μέθοδο ώστε τα experiments να είναι ομοιόμορφα. Βασικά πεδία που χρησιμοποιούμε:

- `l`: στόχος τελικού μήκους διαστήματος,

- `eps`: παράμετρος ε,
- `h`, `grad_tol`: αριθμητική παράγωγος,
- `df`: αναλυτική παράγωγος,
- `param_name`, `values`: η σάρωση των παραμέτρων που κάνουμε στα διάφορα experiments (π.χ. ως προς l).

Στα `ex_*` scripts ορίζουμε ένα *base* `cfg` και για κάθε run αλλάζει μόνο η τρέχουσα τιμή της παραμέτρου (π.χ. l), όλα τα υπόλοιπα μένουν ίδια.