

Αριθμητική Ανάλυση 1η Υποχρεωτική Εργασία

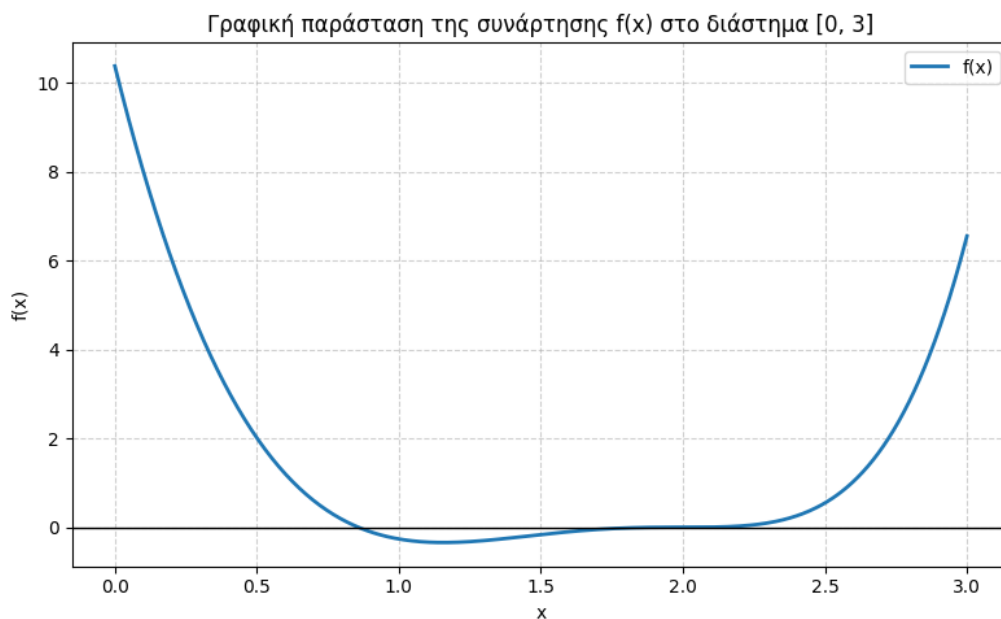
Κωνσταντίνος Δεβετζίδης
ΑΕΜ: 3708

Δεκέμβριος 2025

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Σχολή Θετικών Επιστημών
Τμήμα Πληροφορικής

1 Προσέγγιση Ριζών

Υπολογισμός ριζών της $f(x) = 14x^{x-2} - 12x^{-2} - 7x^3 + 20x^2 - 26x + 12$ με ακρίβεια 6ου δεκαδικού ψηφίου, με τρεις διαφορετικές μεθόδους.



Σχήμα 1: $f(x) = 14x^{x-2} - 12x^{-2} - 7x^3 + 20x^2 - 26x + 12$

Υπολογίζουμε τη γραφική παράσταση της συνάρτησης $f(x)$ στο διάστημα $[0, 3]$ με χρήση της βιβλιοθήκης matplotlib της Python (*plot_function.py*). Από το γράφημα παρατηρούμε ότι η συνάρτηση τέμνει τον άξονα x περίπου στα σημεία $x = 0.8$ και $x = 2$. Αυτό μας δίνει μια πρώτη εκτίμηση για τις ρίζες και μας βοηθά να διαλέξουμε κατάλληλα αρχικά διαστήματα/τιμές για τις αριθμητικές μεθόδους.

1.1 Διχοτόμηση

Περιγραφή Αλγορίθμου

Η μέθοδος της διχοτόμησης εφαρμόζεται όταν η συνάρτηση $f(x)$ αλλάζει πρόσημο σε ένα διάστημα $[a, b]$, δηλαδή όταν $f(a)f(b) < 0$. Σε κάθε βήμα υπολογίζεται το μέσο του διαστήματος, $c = \frac{a+b}{2}$ και εξετάζεται το πρόσημο του $f(c)$. Αν το $f(c)$ είναι μηδέν, τότε το σημείο αυτό θεωρείται ρίζα και η μέθοδος τερματίζει. Διαφορετικά, επιλέγεται το υποδιάστημα στο οποίο συνεχίζει να υπάρχει αλλαγή πρόσημου και η διαδικασία επαναλαμβάνεται. Υπάρχουν δύο κριτήρια τερματισμού:

1. **Ακριβής μηδενισμός:** αν $|f(c)| < \epsilon$ μηχανής, το c είναι ουσιαστικά η ρίζα.
2. **Ακρίβεια στο x :** αν το μισό μήκος του διαστήματος $\frac{b-a}{2}$ γίνει μικρότερο από την ανοχή tol , τότε το c αποτελεί προσεγγιστική ρίζα με ζητούμενη ακρίβεια.

Περιγραφή κώδικα

Η συνάρτηση `bisection(f, a, b, tol, maxit)` που βρίσκεται στο `bisection.py` υλοποιεί τον παραπάνω αλγόριθμο. Ελέγχει αρχικά αν η συνάρτηση αλλάζει πρόσημο. Σε κάθε επανάληψη υπολογίζει το `midpoint c`. Αν το $f(c)$ μηδενίζεται, τερματίζει αμέσως. Αν το διάστημα έχει μικρύνει αρκετά, επιστρέφει την τρέχουσα προσεγγιστική ρίζα. Διαφορετικά, επιλέγει το υποδιάστημα που εξακολουθεί να περιέχει τη ρίζα. Το αποτέλεσμα είναι μια προσεγγιστική ρίζα και ο αριθμός επαναλήψεων.

Στο `bisection_demo.py` παρουσιάζεται η λειτουργία του αλγορίθμου για την εύρεση των ριζών της άσκησης. Αρχικά, ορίζεται η συνάρτηση $f(x)$, στη δική μας περίπτωση, $f(x) = 14x^{x-2} - 12x^{-2} - 7x^3 + 20x^2 - 26x + 12$. Καθορίζεται το κατώφλι ακρίβειας ($tol=1e-6$) και ο μέγιστος αριθμός επαναλήψεων (`maxit`) και δημιουργούνται λίστες με δέκα διαφορετικά διαστήματα γύρω από τις δύο ρίζες της εξίσωσης $f(x) = 0$. Για κάθε διάστημα, το πρόγραμμα καλεί τη συνάρτηση `bisection` και τυπώνει στην κονσόλα το διάστημα που χρησιμοποιήθηκε, την προσεγγιστική ρίζα και τον αριθμό επαναλήψεων.

```

---- Διχοτόμηση για ρίζα κοντά στο 0.8 ----
Δοκιμή 1: διάστημα [0.60, 1.20] --> Ρίζα = 0.857142, Επαναλήψεις = 20
Δοκιμή 2: διάστημα [0.70, 1.10] --> Ρίζα = 0.857143, Επαναλήψεις = 19
Δοκιμή 3: διάστημα [0.50, 1.30] --> Ρίζα = 0.857143, Επαναλήψεις = 20
Δοκιμή 4: διάστημα [0.75, 1.00] --> Ρίζα = 0.857142, Επαναλήψεις = 18
Δοκιμή 5: διάστημα [0.80, 1.20] --> Ρίζα = 0.857143, Επαναλήψεις = 19
Δοκιμή 6: διάστημα [0.65, 1.15] --> Ρίζα = 0.857143, Επαναλήψεις = 19
Δοκιμή 7: διάστημα [0.55, 1.25] --> Ρίζα = 0.857143, Επαναλήψεις = 20
Δοκιμή 8: διάστημα [0.72, 1.05] --> Ρίζα = 0.857142, Επαναλήψεις = 19
Δοκιμή 9: διάστημα [0.78, 1.22] --> Ρίζα = 0.857143, Επαναλήψεις = 19
Δοκιμή 10: διάστημα [0.68, 1.18] --> Ρίζα = 0.857142, Επαναλήψεις = 19

---- Διχοτόμηση για ρίζα κοντά στο 2.000 ----
Δοκιμή 1: διάστημα [1.80, 2.90] --> Ρίζα = 1.999989, Επαναλήψεις = 16
Δοκιμή 2: διάστημα [1.70, 2.50] --> Ρίζα = 2.000000, Επαναλήψεις = 3
Δοκιμή 3: διάστημα [1.60, 2.40] --> Ρίζα = 2.000000, Επαναλήψεις = 1
Δοκιμή 4: διάστημα [1.90, 2.30] --> Ρίζα = 1.999988, Επαναλήψεις = 15
Δοκιμή 5: διάστημα [1.75, 2.20] --> Ρίζα = 2.000008, Επαναλήψεις = 15
Δοκιμή 6: διάστημα [1.85, 2.40] --> Ρίζα = 2.000000, Επαναλήψεις = 17
Δοκιμή 7: διάστημα [1.65, 2.35] --> Ρίζα = 2.000000, Επαναλήψεις = 1
Δοκιμή 8: διάστημα [1.80, 2.20] --> Ρίζα = 2.000000, Επαναλήψεις = 1
Δοκιμή 9: διάστημα [1.70, 2.10] --> Ρίζα = 2.000000, Επαναλήψεις = 2
Δοκιμή 10: διάστημα [1.60, 2.00] --> Ρίζα = 1.999988, Επαναλήψεις = 15

```

Σχήμα 2: Αποτελέσματα bisection_demo

1.2 Newton-Raphson

Περιγραφή αλγορίθμου

Η μέθοδος Newton-Raphson χρησιμοποιεί την παράγωγο της συνάρτησης και βασίζεται στην προσέγγιση της καμπύλης της $f(x)$ με την εφαπτομένη της σε κάθε βήμα. Ο αναδρομικός τύπος που δίνει την επόμενη προσέγγιση είναι x_{n+1} είναι:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Η μέθοδος συγκλίνει συνήθως πολύ ταχύτερα από αυτή της διχοτόμησης υπό την προϋπόθεση ότι η αρχική εκτίμηση είναι κοντά στη ρίζα και η $f'(x)$ δεν μηδενίζεται στη ρίζα.

Περιγραφή κώδικα

Η συνάρτηση `newton(f, df, x0, tol, maxit)`, βρίσκεται στο αρχείο `newton.py` και δέχεται ως ορίσματα τη συνάρτηση f , την παράγωγο της f (df), την αρχική τιμή x_0 , το κατώφλι ακρίβειας (tol) και τον μέγιστο αριθμό επαναλήψεων ($maxit$). Σε κάθε επανάληψη υπολογίζει το νέο x και ελέγχει αν η διαφορά $|x_n - x_{n-1}|$ είναι μικρότερη από το tol . Στο `asll_demo.py` ορίζονται η $f(x)$ και $f'(x)$ και εκτελείται ο αλγόριθμος για 10 διαφορετικές αρχικές τιμές γύρω από κάθε ρίζα.

Αποτελέσματα και Σύγκλιση

Παρατηρώντας τα αποτελέσματα του `newton_demo.py` στο Σχήμα 3 παρακάτω, βλέπουμε ότι για τη ρίζα κοντά στο 0.8, η μέθοδος εμφανίζει εξαιρετικά γρήγορη σύγκλιση. Απαιτούνται λίγες επαναλήψεις (4-8) για να επιτευχθεί ακρίβεια 6 δεκαδικών ψηφίων. Για τη δεύτερη ρίζα, παρατηρούμε ότι ο αριθμός των επαναλήψεων είναι σημαντικά αυξημένος σε σχέση με την πρώτη, παρόλο που η ακρίβεια παραμένει η ίδια (10^{-6}). Επομένως, η μέθοδος Newton-Raphson συγκλίνει τετραγωνικά για τη ρίζα $x_1 = 0.857143$, ενώ χάνει την τετραγωνική της σύγκλιση για την ρίζα $x_2 = 2$. Αυτό συμβαίνει διότι η ταχύτητα σύγκλισης της μεθόδου εξαρτάται από το αν η ρίζα είναι απλή ή πολλαπλή. Στο x_1 , η $f'(x_1)$ είναι διάφορη του μηδενός (απλή ρίζα). Στο x_2 , η $f'(x_2)$ μηδενίζεται (πολλαπλή ρίζα). Στις πολλαπλές ρίζες, η μέθοδος Newton-Raphson έχει γραμμική σύγκλιση, γεγονός που εξηγεί τον μεγάλο αριθμό επαναλήψεων που παρατηρούμε στα αποτελέσματα.

```

---- Newton-Raphson για ρίζα κοντά στο 0.8 ----
Δοκιμή 1: x0 = 0.2 --> ρίζα = 0.857143, επαναλήψεις = 7
Δοκιμή 2: x0 = 0.3 --> ρίζα = 0.857143, επαναλήψεις = 6
Δοκιμή 3: x0 = 0.4 --> ρίζα = 0.857143, επαναλήψεις = 6
Δοκιμή 4: x0 = 0.5 --> ρίζα = 0.857143, επαναλήψεις = 6
Δοκιμή 5: x0 = 0.6 --> ρίζα = 0.857143, επαναλήψεις = 5
Δοκιμή 6: x0 = 0.7 --> ρίζα = 0.857143, επαναλήψεις = 5
Δοκιμή 7: x0 = 0.8 --> ρίζα = 0.857143, επαναλήψεις = 4
Δοκιμή 8: x0 = 0.9 --> ρίζα = 0.857143, επαναλήψεις = 4
Δοκιμή 9: x0 = 1.0 --> ρίζα = 0.857143, επαναλήψεις = 5
Δοκιμή 10: x0 = 1.1 --> ρίζα = 0.857143, επαναλήψεις = 8

---- Newton-Raphson για ρίζα κοντά στο 2 ----
Δοκιμή 1: x0 = 1.5 --> ρίζα = 1.999993, επαναλήψεις = 27
Δοκιμή 2: x0 = 1.6 --> ρίζα = 1.999986, επαναλήψεις = 33
Δοκιμή 3: x0 = 1.7 --> ρίζα = 1.999994, επαναλήψεις = 33
Δοκιμή 4: x0 = 1.8 --> ρίζα = 1.999989, επαναλήψεις = 25
Δοκιμή 5: x0 = 1.9 --> ρίζα = 1.999997, επαναλήψεις = 26
Δοκιμή 6: x0 = 2.1 --> ρίζα = 1.999998, επαναλήψεις = 24
Δοκιμή 7: x0 = 2.2 --> ρίζα = 1.999991, επαναλήψεις = 35
Δοκιμή 8: x0 = 2.3 --> ρίζα = 2.000012, επαναλήψεις = 26
Δοκιμή 9: x0 = 2.4 --> ρίζα = 2.000011, επαναλήψεις = 27
Δοκιμή 10: x0 = 2.5 --> ρίζα = 2.000013, επαναλήψεις = 34

```

Σχήμα 3: Αποτελέσματα newton_demo

1.3 Τέμνουσα (Secant)

Περιγραφή αλγορίθμου

Η μέθοδος της τέμνουσας αποτελεί μια παραλλαγή της μεθόδου Newton-Raphson, η οποία αποφεύγει τον υπολογισμό της $f'(x)$. Αντί για την εφαπτομένη, χρησιμοποιεί μια τέμνουσα ευθεία που διέρχεται από δύο προηγούμενα σημεία x_{n-1} και x_n για να εκτιμήσει την επόμενη θέση της ρίζας. Ο αναδρομικός τύπος είναι:

$$x_{n+1} = x_n - f(x_n) \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

Η μέθοδος χρειάζεται δύο αρχικές τιμές, x_0 και x_1 , ενώ συγκλίνει ταχύτερα από τη διχοτόμηση και ελαφρώς πιο αργά από τη Newton-Raphson.

Περιγραφή κώδικα

Στο `secant.py` υλοποιείται η συνάρτηση `secant(f, x0, x1, tol, maxit)` όπου x_0 και x_1 τα αρχικά σημεία εκκίνησης. Σε κάθε βήμα υπολογίζει το νέο σημείο τομής της τέμνουσας με τον άξονα των x και ελέγχει τη διαφορά $f(x_n) - f(x_{n-1})$ για να αποφεύγεται η διαίρεση με το μηδέν. Στο `ask1_demo.py` εκτελούνται 10 δοκιμές για κάθε ρίζα, επιλέγοντας ζεύγη αρχικών τιμών κοντά στις περιοχές ενδιαφέροντος.

Αποτελέσματα

Παρατηρώντας τα αποτελέσματα στο παρακάτω σχήμα, βλέπουμε ότι η μέθοδος συγκλίνει εξαιρετικά γρήγορα για τη ρίζα κοντά στο 0.8, καθώς χρειάστηκε μόλις 5-9 επαναλήψεις για να πετύχει την επιθυμητή ακρίβεια. Για τη ρίζα κοντά στο 2 υπάρχει μεγάλη αύξηση στο πλήθος των επαναλήψεων (24-42). Αυτό συμβαίνει γιατί η ρίζα $x_2 = 2$ είναι πολλαπλή ρίζα. Στις πολλαπλές ρίζες η παράγωγος μηδενίζεται και η τάξη σύγκλισης της μεθόδου μειώνεται. Μάλιστα σε ορισμένες δοκιμές ο αλγόριθμος δε μπόρεσε να τερματίσει λόγω διαίρεσης με το 0. Αυτό εξηγείται από τον τύπο της Τέμνουσας. Κοντά σε μια πολλαπλή ρίζα, η καμπύλη της συνάρτησης γίνεται επίπεδη. Αυτό σημαίνει ότι για δύο κοντινά σημεία x_n, x_{n-1} , οι τιμές $f(x_n)$ και $f(x_{n-1})$ είναι σχεδόν ίσες, οδηγώντας τον παρονομαστή $f(x_n) - f(x_{n-1})$ να γίνει μηδέν και προκαλώντας διαίρεση με το μηδέν και κατάρρευση του αλγορίθμου.

```

---- Μέθοδος Τέμνουσας για ρίζα κοντά στο 0.8 ----
Δοκιμή 1: x0=0.7, x1=0.8 --> Ρίζα = 0.857143, Επαναλήψεις = 5
Δοκιμή 2: x0=0.8, x1=0.9 --> Ρίζα = 0.857143, Επαναλήψεις = 5
Δοκιμή 3: x0=0.6, x1=0.7 --> Ρίζα = 0.857143, Επαναλήψεις = 6
Δοκιμή 4: x0=0.9, x1=1.0 --> Ρίζα = 0.857143, Επαναλήψεις = 6
Δοκιμή 5: x0=0.5, x1=0.6 --> Ρίζα = 0.857143, Επαναλήψεις = 7
Δοκιμή 6: x0=1.0, x1=1.1 --> Ρίζα = 0.857143, Επαναλήψεις = 9
Δοκιμή 7: x0=0.75, x1=0.85 --> Ρίζα = 0.857143, Επαναλήψεις = 4
Δοκιμή 8: x0=0.85, x1=0.95 --> Ρίζα = 0.857143, Επαναλήψεις = 5
Δοκιμή 9: x0=0.65, x1=0.75 --> Ρίζα = 0.857143, Επαναλήψεις = 6
Δοκιμή 10: x0=0.8, x1=1.0 --> Ρίζα = 0.857143, Επαναλήψεις = 6

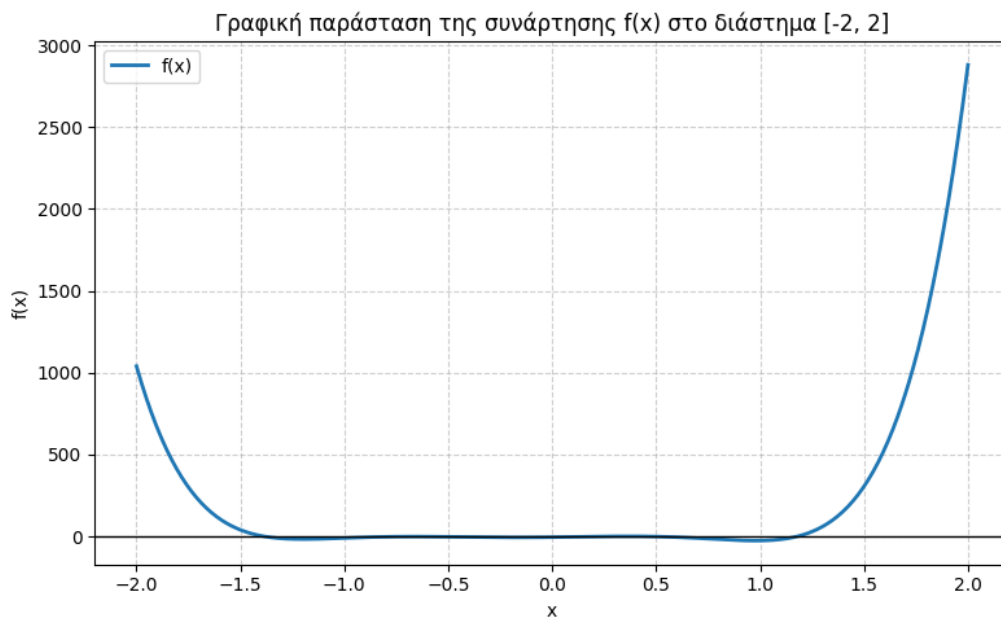
---- Μέθοδος Τέμνουσας για ρίζα κοντά στο 2.0 ----
Δοκιμή 1: x0=1.8, x1=1.9 --> Ρίζα = 1.999986, Επαναλήψεις = 33
Δοκιμή 2: x0=2.1, x1=2.2 --> Ρίζα = 2.000011, Επαναλήψεις = 33
Δοκιμή 3: x0=1.9, x1=2.1 --> Ρίζα = 2.000000, Επαναλήψεις = 29
Δοκιμή 4: x0=1.7, x1=1.8 --> Ρίζα = 1.999989, Επαναλήψεις = 35
Δοκιμή 5: x0=2.2, x1=2.3 --> Ρίζα = 2.000007, Επαναλήψεις = 39
Τερματισμός. Παρονομαστής=0. f(x_n)==f(x_n-1)
Δοκιμή 6: x0=1.5, x1=1.6 --> Ρίζα = 1.999990, Επαναλήψεις = 37
Τερματισμός. Παρονομαστής=0. f(x_n)==f(x_n-1)
Δοκιμή 7: x0=2.4, x1=2.5 --> Ρίζα = 2.000017, Επαναλήψεις = 38
Δοκιμή 8: x0=1.8, x1=2.2 --> Ρίζα = 1.999989, Επαναλήψεις = 31
Τερματισμός. Παρονομαστής=0. f(x_n)==f(x_n-1)
Δοκιμή 9: x0=1.95, x1=2.05 --> Ρίζα = 1.999992, Επαναλήψεις = 24
Δοκιμή 10: x0=1.6, x1=1.7 --> Ρίζα = 1.999990, Επαναλήψεις = 42

```

Σχήμα 4: Αποτελέσματα μεθόδου τέμνουσας

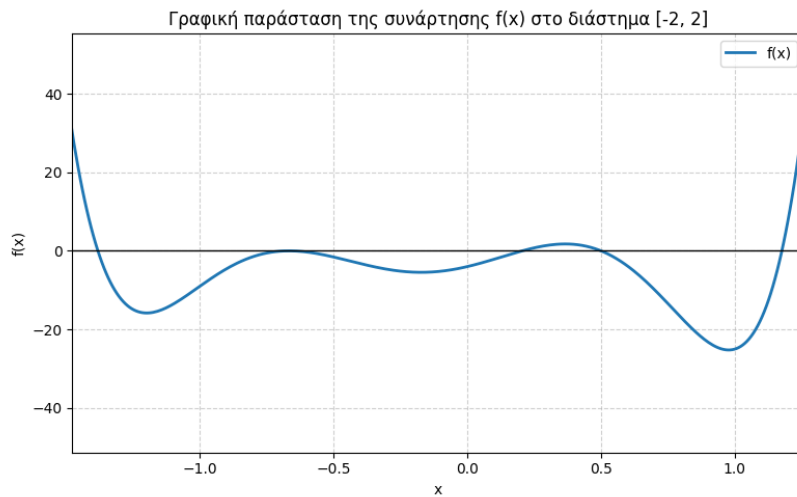
2 Τροποποιημένη Προσέγγιση Ριζών

2.1 Εύρεση ριζών συνάρτησης με τις τροποποιημένες μεθόδους



Σχήμα 5: $f(x) = 54x^6 + 45x^5 - 102x^4 - 69x^3 + 35x^2 + 16x - 4$

Μεγεθύνοντας τη γραφική παράσταση (Σχήμα 6), βρίσκουμε κοντά σε ποια σημεία/διαστήματα να αναζητήσουμε τις ρίζες της $f(x)$, τα οποία είναι: -1.3 , -0.6 , 0.3 , 0.5 , 1.1 και δίνονται ως αρχικά σημεία για την τροποποιημένη Newton-Raphson. Για την τροποποιημένη διχοτόμηση ορίζουμε αρχικά διαστήματα μήκους 0.4 γύρω από κάθε πιθανή ρίζα. Π.χ. για ρίζα κοντά στο -1.3 , αρχικό διάστημα $[-1.5, -1.1]$. Για την τροποποιημένη μέθοδο της τέμνουσας, δίνουμε τα τρία κοντινότερα σημεία με ακρίβεια ενός δεκαδικού ψηφίου, ως τα τρία αρχικά σημεία. Π.χ. για ρίζα κοντά στο -1.3 , αρχικά σημεία -1.4 , -1.3 , -1.2 .



Σχήμα 6: Μεγέθυνση κοντά στον άξονα των x

```

--- Αναζήτηση κοντά στο -1.3 ---
Τροπ. Διχοτόμηση: Ρίζα=-1.3812985, Επαναλήψεις=17
Τροπ. Newton : Ρίζα=-1.3812985, Επαναλήψεις=4
Τροπ. Τέμνουσα : Ρίζα=-1.3812985, Επαναλήψεις=7

--- Αναζήτηση κοντά στο -0.6 ---
Τροπ. Διχοτόμηση: Απέτυχε (μη αλλαγή προσήμου)
Τροπ. Newton : Ρίζα=-0.6666666, Επαναλήψεις=13
Τροπ. Τέμνουσα : Ρίζα=-0.6666667, Επαναλήψεις=24

--- Αναζήτηση κοντά στο 0.3 ---
Τροπ. Διχοτόμηση: Ρίζα=0.2051829, Επαναλήψεις=18
Τροπ. Newton : Ρίζα=0.2051829, Επαναλήψεις=4
Τροπ. Τέμνουσα : Ρίζα=0.2051829, Επαναλήψεις=4

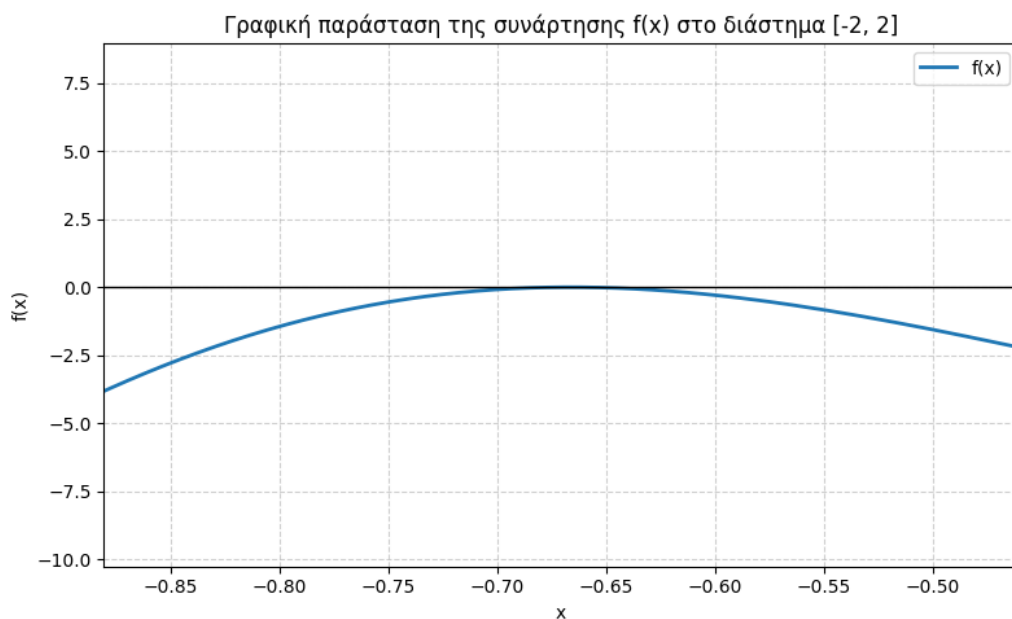
--- Αναζήτηση κοντά στο 0.5 ---
Τροπ. Διχοτόμηση: Ρίζα=0.5000000, Επαναλήψεις=15
Τροπ. Newton : Ρίζα=0.5000000, Επαναλήψεις=1
Τροπ. Τέμνουσα : Ρίζα=0.6000000, Επαναλήψεις=1

--- Αναζήτηση κοντά στο 1.1 ---
Τροπ. Διχοτόμηση: Ρίζα=1.1761155, Επαναλήψεις=17
Τροπ. Newton : Ρίζα=1.1761156, Επαναλήψεις=4
Τροπ. Τέμνουσα : Ρίζα=1.1761156, Επαναλήψεις=5

```

Σχήμα 7: Αποτελέσματα αναζήτησης ριζών

Παρατηρούμε ότι οι τροποποιημένες μέθοδοι Τέμνουσας και Newton-Raphson συγκλίνουν πολύ γρηγορότερα από αυτή της Διχοτόμησης, η οποία δεν μπόρεσε να ολοκληρωθεί στην περίπτωση της αναζήτησης ρίζας κοντά στο -0.6 . Αυτό συμβαίνει γιατί η $f(x)$ έχει ρίζα κοντά στο 0.6 , αλλά δεν αλλάζει πρόσημο γύρω από αυτή, επομένως χωρίς το βασικό κριτήριο $f(a)f(b) < 0$, ο αλγόριθμος δεν μπορεί να εκτελεστεί για $a = -0.8$ και $b = -0.4$. Αυτό φαίνεται καθαρά και στο παρακάτω σχήμα.



Σχήμα 8: $f(x)$ κοντά στο -0.6

2.2 Μελέτη σύγκλισης της διχοτόμησης

Για να διερευνήσουμε τη συμπεριφορά της τροποποιημένης μεθόδου διχοτόμησης, εκτελούμε τον αλγόριθμο 20 φορές, μεταβάλλοντας τυχαία τα άκρα του διαστήματος στο οποίο αναζητούμε τη ρίζα σε κάθε εκτέλεση. Θα αναζητήσουμε τη ρίζα κοντά στο $-1, 3$.

Δοκιμή	Διάστημα	Ρίζα	Επαναλήψεις
1	[-1.7161, -1.2012]	-1.3812985	17
2	[-1.7532, -1.1536]	-1.3812984	19
3	[-1.8015, -1.2220]	-1.3812985	18
4	[-1.7203, -1.1545]	-1.3812985	18
5	[-1.7520, -1.2099]	-1.3812985	19
6	[-1.8390, -1.2383]	-1.3812985	19
7	[-1.8175, -1.2056]	-1.3812985	19
8	[-1.7670, -1.1963]	-1.3812985	17
9	[-1.7776, -1.2471]	-1.3812985	17
10	[-1.7737, -1.1595]	-1.3812985	20
11	[-1.8110, -1.2088]	-1.3812985	18
12	[-1.8445, -1.1564]	-1.3812985	17
13	[-1.7205, -1.1827]	-1.3812985	18
14	[-1.8882, -1.1533]	-1.3812985	18
15	[-1.8986, -1.1988]	-1.3812985	18
16	[-1.8166, -1.2181]	-1.3812985	19
17	[-1.8827, -1.2451]	-1.3812985	19
18	[-1.8904, -1.1768]	-1.3812985	19
19	[-1.8672, -1.1634]	-1.3812985	17
20	[-1.7767, -1.2462]	-1.3812984	17

Σχήμα 9: Αποτελέσματα δοκιμών

Παρατηρούμε ότι ο αριθμός των επαναλήψεων δεν είναι σταθερός, αλλά κυμαίνεται από 17 σε 20. Αυτό συμβαίνει διότι η τροποποιημένη διχοτόμηση κόβει το διάστημα στο $\frac{1}{3}$ και όχι στο $\frac{1}{2}$, οπότε η ταχύτητα σύγκλισης εξαρτάται από τη σχετική θέση της ρίζας μέσα στο αρχικό διάστημα. Δηλαδή, αν η ρίζα τυχαίνει να βρίσκεται στο "μεγάλο" κομμάτι ($\frac{2}{3}$) που κρατάει ο αλγόριθμος σε ένα βήμα, η σύγκλιση καθυστερεί, ενώ αν βρίσκεται στο "μικρό" κομμάτι ($\frac{1}{3}$), επιταχύνεται. Οι μικρές διαφορές στο τελευταίο δεκαδικό ψηφίο της ρίζας, οφείλονται στη στρωγυλοποίηση που κάνει ο αλγόριθμος όταν έχει πετύχει την επιθυμητή ακρίβεια των 7 δεκαδικών ψηφίων (από ποια πλευρά προσεγγίζει τη ρίζα).

2.3 Σύγκριση ταχύτητας

Για να συγκρίνουμε την ταχύτητα σύγκλισης, εκτελούμε κάθε ζεύγος αλγορίθμων στην κλασσική και στην τροποποιημένη του υλοποίηση, 100 φορές. Σε κάθε εκτέλεση χρησιμοποιούμε τυχαίες αρχικές τιμές γύρω από την περιοχή κοντά στο -1.3 και καταγράφουμε το πλήθος των επαναλήψεων που απαιτήθηκαν. Τέλος, υπολογίζουμε το μέσο όρο των επαναλήψεων.

Κλασσική Newton-Ralphson:	8.33
Τροπ/μένη Newton-Ralphson:	5.99
Κλασσική Διχοτόμηση:	22.95
Τροπ/μένη Διχοτόμηση:	18.74
Κλασσική Τέμνουσα:	6.63
Τροπ/μένη Τέμνουσα:	10.30

Σχήμα 10: Αποτελέσματα σύγκρισης ταχύτητας

Η τροποποιημένη μέθοδος Newton-Raphson, χρησιμοποιώντας και τη δεύτερη παράγωγο, επιτυγχάνει ταχύτερη σύγκλιση από την κλασσική υλοποίησή της. Παρομοίως, η τροποποιημένη διχοτόμηση αποδίδει καλύτερα από την κλασσική, βασισμένη στο ποιο άκρο είναι πιο κοντά στη ρίζα. Παρατηρούμε ότι η τροποποιημένη τέμνουσα χρειάζεται περισσότερες επαναλήψεις από την κλασσική. Αυτό συμβαίνει γιατί η κλασσική μέθοδος προσεγγίζει τη συνάρτηση με μια ευθεία γραμμή, ενώ η τροποποιημένη μέθοδος προσπαθεί να προσαρμόζει μια παραβολή περνώντας από 3 τυχαία σημεία. Λόγω της τυχαίας αρχικοποίησης, τα 3 αυτά σημεία μπορεί να σχηματίσουν μια παραβολή που εκτοξεύει την επόμενη εκτίμηση πολύ μακριά από τη ρίζα (overshooting). Επιπλέον, ο τύπος της τροποποιημένης μεθόδου έχει παρονομαστή τον όρο $(q - 1)(r - 1)(s - 1)$. Όταν τα αρχικά σημεία είναι κοντά μεταξύ τους, ο παρονομαστής τείνει στο μηδέν και προκαλείται αριθμητική αστάθεια. (Χρησιμοποιήθηκε Gemini για τον έλεγχο της σωστής λειτουργίας της τροποποιημένης τέμνουσας και την επεξήγηση της καθυστέρησης στην ταχύτητα).

Ανοίγοντας το `ask2_demo.py` μπορείτε να τρέξετε όλα τα πειράματα που πραγματοποιήθηκαν για την Άσκηση 2.

3 Επίλυση Γραμμικών Συστημάτων

3.1 PA=LU decomposition και επίλυση Γραμμικού Συστήματος

Οι δύο συναρτήσεις που υλοποιήθηκαν για αυτό το ερώτημα είναι οι $lu_dec(A)$ και $solve_lu(P, L, U, b)$ οι οποίες βρίσκονται στο αρχείο *ask3_lib.py*.

Ανάλυση κώδικα

Η συνάρτηση $lu_dec(A)$ δέχεται έναν τετραγωνικό πίνακα A και επιστρέφει τους πίνακες P , L και U ακολουθώντας τα εξής βήματα:

1. Αρχικοποίηση

- Ο πίνακας U αρχικοποιείται ως αντίγραφο του A . Σταδιακά θα μετατραπεί σε άνω τριγωνικό μέσω της απαλοιφής.
- Ο πίνακας L αρχικοποιείται με μηδενικά. Εκεί θα αποθηκεύονται οι πολλαπλασιαστές της απαλοιφής Gauss.
- Ο πίνακας P αρχικοποιείται ως μοναδιαίος πίνακας και καταγράφει τις εναλλαγές γραμμών.

2. Μερική Οδήγηση

Για κάθε στήλη k , εντοπίζουμε το στοιχείο με τη μέγιστη απόλυτη τιμή από τη διαγώνιο και κάτω ($np.argmax(np.abs(U[k :, k]))$). Αν το μέγιστο στοιχείο δεν βρίσκεται στην τρέχουσα γραμμή, κάνουμε αντιμετάθεση γραμμών στους πίνακες U , P και L . Κάνοντας χρήση των διπλών brackets που μας παρέχει η Python ($L[[k, max_index]] = L[[max_index, k]]$) διασφαλίζουμε ότι οι μετακινήσεις γίνονται εύκολα και σωστά.

3. Απαλοιφή Gauss

Για κάθε γραμμή i κάτω από τη διαγώνιο ($i > k$), υπολογίζουμε τον πολλαπλασιαστή $rol = U_{ik}/U_{kk}$, τον αποθηκεύουμε στον πίνακα L και αφαιρούμε την γραμμή-οδηγό (πολλαπλασιασμένη με τον rol) από την τρέχουσα γραμμή του U , μηδενίζοντας το στοιχείο U_{ik} .

Στο τέλος, προσθέτουμε τη μονάδα στην κύρια διαγώνιο του L ($np.fill_diagonal(L, 1)$). Προτίμησα να προσθέσω τη μονάδα στη διαγώνιο του L στο τέλος, αντί να τον

αρχικοποιήσω και αυτόν ως μοναδιαίο, για να κάνω πιο εύκολα τις αλλαγές γραμμών.

Η συνάρτηση $\text{solve_lu}(P, L, U, b)$ επιλύει το σύστημα $Ax = b$ αξιοποιώντας την παραγοντοποίηση. Η εξίσωση $Ax = b$ μετατρέπεται σε $PAx = Pb \Rightarrow LUX = Pb$. Η επίλυση γίνεται σε τρία στάδια:

1. Αναδιάταξη του b
Υπολογίζουμε το διάνυσμα $Pb = P \cdot b$. Αυτό αναδιατάσσει τους όρους του b σύμφωνα με τις εναλλαγές γραμμών που έγιναν κατά την παραγοντοποίηση.
2. Εμπρός Αντικατάσταση
Λύνουμε το σύστημα $Ly = Pb$. Επειδή ο L είναι κάτω τριγωνικός, υπολογίζουμε τα y_i από το πρώτο έως το τελευταίο.
3. Πίσω Αντικατάσταση
Λύνουμε το σύστημα $Ux = y$. Επειδή ο U είναι άνω τριγωνικός, υπολογίζουμε τα x_i από το τελευταίο προς το πρώτο.

Αποτελέσματα Εκτέλεσης

Στο πρώτο μέρος του `ask3_demo`, εφαρμόζουμε τους παραπάνω αλγορίθμους. Για δοκιμή δίνουμε τους πίνακες

$$A = \begin{bmatrix} 2 & 1 & 1 \\ 4 & 3 & 3 \\ 8 & 7 & 9 \end{bmatrix}, b = \begin{bmatrix} 4 \\ 10 \\ 24 \end{bmatrix}$$

Η εκτέλεση του κώδικα παρήγαγε τους εξής πίνακες:

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, L = \begin{bmatrix} 1 & 0 & 0 \\ 0.25 & 1 & 0 \\ 0.5 & 0.66\dots & 1 \end{bmatrix}, U = \begin{bmatrix} 8 & 7 & 9 \\ 0 & -0.75 & -1.25 \\ 0 & 0 & -0.66\dots \end{bmatrix}$$

Τελική Λύση: Η επίλυση του συστήματος έδωσε το ακριβές αποτέλεσμα:

$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

```

Πίνακας A:
[[2. 1. 1.]
 [4. 3. 3.]
 [8. 7. 9.]]

P:
[[0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]]

L:
[[1.      0.      0.      ]
 [0.25    1.      0.      ]
 [0.5     0.66666667 1.      ]]

U:
[[ 8.      7.      9.      ]
 [ 0.     -0.75    -1.25    ]
 [ 0.      0.     -0.66666667]]

Λύση συστήματος (LU): x = [1. 1. 1.]

```

Σχήμα 11: Δοκιμή PA=LU

3.2 Αποσύνθεση Cholesky

Για το συγκεκριμένο ερώτημα υλοποιήθηκε η συνάρτηση $cholesky(A)$ στο αρχείο `ask3_lib.py`.

Ανάλυση Κώδικα

Η συνάρτηση $cholesky(A)$ δέχεται έναν *συμμετρικό και θερικά ορισμένο* πίνακα A και επιστρέφει τον κάτω τριγωνικό πίνακα L . Η διαδικασία έχει ως εξής:

1. Αρχικοποίηση
Αρχικοποιούμε τον πίνακα L με μηδενικά.
2. Διπλός Βρόχος Υπολογισμού
Διατρέχουμε τον πίνακα γραμμή προς γραμμή (i από 0 έως $n - 1$) και για κάθε γραμμή υπολογίζουμε τα στοιχεία μέχρι τη διαγώνιο (j από 0 έως i).
3. Υπολογισμός Στοιχείων
Για κάθε στοιχείο L_{ij} , υπολογίζουμε πρώτα το άθροισμα των γινομένων των προηγούμενων στοιχείων της γραμμής i και της γραμμής j (που αντιστοιχεί στη στήλη j του L): $sum_val = np.dot(L[i, : j], L[j, : j])$. Υπάρχουν δύο περιπτώσεις:

- Διαγώνια Στοιχεία ($i = j$): Ο τύπος είναι: $L_{ii} = \sqrt{A_{ii} - \sum_{k=0}^{i-1} L_{ik}^2}$.
- Μη Διαγώνια Στοιχεία ($i \neq j$): Ο τύπος είναι: $L_{ij} = \frac{1}{L_{jj}} \left(A_{ij} - \sum_{k=0}^{j-1} L_{ik} L_{jk} \right)$.

Αποτελέσματα Εκτέλεσης

Στο δεύτερο μέρος του `ask3_demo`, εφαρμόζουμε τον παραπάνω αλγόριθμο. Για δοκιμή δίνουμε τον πίνακα:

$$A = \begin{bmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{bmatrix}$$

Η εκτέλεση του κώδικα παρήγαγε τον πίνακα L :

$$L = \begin{bmatrix} 2 & 0 & 0 \\ 6 & 1 & 0 \\ -8 & 5 & 3 \end{bmatrix}$$

Συμμετρικός Πίνακας A:

$\begin{bmatrix} 4. & 12. & -16. \end{bmatrix}$

$\begin{bmatrix} 12. & 37. & -43. \end{bmatrix}$

$\begin{bmatrix} -16. & -43. & 98. \end{bmatrix}$

Πίνακας L (Cholesky):

$\begin{bmatrix} 2. & 0. & 0. \end{bmatrix}$

$\begin{bmatrix} 6. & 1. & 0. \end{bmatrix}$

$\begin{bmatrix} -8. & 5. & 3. \end{bmatrix}$

Σχήμα 12: Δοκιμή Cholesky

3.3 Gauss-Seidel

Για το συγκεκριμένο ερώτημα υλοποιήθηκαν οι συναρτήσεις `create_system(n)` και `gauss_seidel(A, b, x0, tol, maxit)` στο αρχείο `ask3_lib.py`.

Ανάλυση Κώδικα

Η συνάρτηση `create_system(n)` δέχεται σαν όρισμα έναν ακέραιο αριθμό n και επιστρέφει έναν συμμετρικό και διαγώνιο κυρίαρχο πίνακα μεγέθους $n \times n$ και έναν πίνακα b μεγέθους n . Για τον πίνακα A , σύμφωνα με την εκφώνηση ισχύει ότι τα στοιχεία της κύριας διαγωνίου είναι $A_{ii} = 5$ και τα στοιχεία της άνω και κάτω διαγωνίου είναι $A_{i,i+1} = A_{i+1,i} = -2$, ενώ ο b έχει την τιμή 1 σε όλες τις θέσεις, εκτός από την πρώτη και την τελευταία όπου έχει την τιμή 3 ($b = [3, 1, \dots, 1, 3]$). Αυτό επιτυγχάνεται στον κώδικα με τον εξής τρόπο:

1. Αρχικοποιούμε τον πίνακα A με μηδενικά (`np.zeros`) και το διάνυσμα b με μονάδες (`np.ones`).
2. Θέτουμε τα άκρα του b σε 3.
3. Μέσω ενός βρόχου `for`, γεμίζουμε την κύρια διαγώνιο του A με 5 και τις δευτερεύουσες με -2 .

Η συνάρτηση `gauss_seidel(A, b, x0, tol, maxit)` δέχεται ως ορίσματα τον πίνακα A , το διάνυσμα b , το αρχικό διάνυσμα λύσης $x^{(0)}$ το οποίο αρχικοποιείται με μηδενικά, εκτός αν δοθεί διαφορετικά στην κλήση της συνάρτησης, την επιθυμητή ακρίβεια `tol` και το μέγιστο αριθμό επαναλήψεων `maxit`. Επιστρέφει το διάνυσμα x και τον αριθμό επαναλήψεων που εκτελέστηκαν. Αυτό επιτυγχάνεται στον κώδικα με τον εξής τρόπο:

1. Δημιουργείται το διάνυσμα λύσης x (αρχικά μηδενικό, αν δεν δοθεί x_0)
2. Ξεκινά η επανάληψη μέχρι το `maxit`. Κρατάμε ένα αντίγραφο της λύσης από την προηγούμενη επανάληψη για τον υπολογισμό και για τον έλεγχο σύγκλισης ($x_{old} = x.copy()$).
3. Ενημέρωση Αγνώστων γραμμή προς γραμμή
 - $s1 = np.dot(A[i, : i], x[: i])$: Υπολογίζουμε τη συνεισφορά των αγνώστων που βρίσκονται πριν το i , χρησιμοποιώντας το διάνυσμα x , το οποίο περιέχει τις νέες τιμές που μόλις υπολογίστηκαν στην τρέχουσα επανάληψη (χαρακτηριστικό της Gauss-Seidel, διαφορά με Jacobi).

- $s2 = np.dot(A[i, i + 1 :], x_{old}[i + 1 :])$: Υπολογίζουμε τη συνεισφορά των αγνώστων που βρίσκονται μετά το i χρησιμοποιώντας το x_{old} , καθώς οι νέες τιμές δεν έχουν υπολογιστεί ακόμα.
 - $x[i] = (b[i] - s1 - s2)/A[i, i]$: Εφαρμόζουμε τον τύπο επίλυσης ως προς x_i .
4. Έλεγχος Σύγκλισης: Υπολογίζουμε τη μέγιστη διαφορά (άπειρη νόρμα) μεταξύ της νέας και της παλιάς λύσης. ($error = np.max(np.abs(x - x_{old}))$). Αν το σφάλμα είναι μικρότερο από την ανοχή ($tol=1e-5$), η διαδικασία τερματίζει και επιστρέφει τη λύση.

Αποτελέσματα Εκτέλεσης

```

--- Επίλυση για n = 10 ---
Αριθμός Επαναλήψεων: 23
Λύση x: [0.99999119 0.99998699 0.99998601 0.99998705 0.99998916 0.99999167
0.99999411 0.99999624 0.99999794 0.99999917]

--- Επίλυση για n = 5000 ---
Αριθμός Επαναλήψεων: 27
Λύση x (πρώτα 5): [0.99999402 0.99999004 0.99998741 0.99998567 0.99998452]
Λύση x (τελευταία 5): [0.99999802 0.99999868 0.99999919 0.99999956 0.99999982]

```

Σχήμα 13: Δοκιμή Gauss-Seidel

Τα αποτελέσματα που πήραμε βρίσκονται εντός του επιθυμητού ορίου ακρίβειας, επομένως η λύση είναι επιτυχής. Για $n = 10$, απαιτήθηκαν 23 επαναλήψεις, ενώ για $n = 5000$, απαιτήθηκαν 27 επαναλήψεις. Αυτό αποδεικνύει ότι η μέθοδος Gauss-Seidel είναι εξαιρετικά αποδοτική για το συγκεκριμένο πρόβλημα. Η σταθερότητα αυτή οφείλεται στο γεγονός ότι ο πίνακας A είναι διαγώνια κυρίαρχος, το οποίο εγγυάται γρήγορη σύγκλιση, αναξάρτητα από το μέγεθος n του πίνακα.

Εκτελώντας το αρχείο `ask3_demo.py` μπορείτε να δείτε όλες τις δοκιμές που έγιναν για την Άσκηση 3.

4 Pagerank

4.1 Απόδειξη στοχαστικότητας

Ο πίνακας Google G ορίζεται ως εξής:

$$G_{ij} = \frac{q}{n} + (1 - q) \frac{A_{ji}}{n_j}$$

Για να είναι ο πίνακας στοχαστικός, πρέπει το άθροισμα των στοιχείων κάθε στήλης j να ισούται με 1. Αθροίζουμε τα στοιχεία της στήλης j :

$$\sum_{i=1}^n G_{ij} = \sum_{i=1}^n \left(\frac{q}{n} + (1 - q) \frac{A_{ji}}{n_j} \right) = \sum_{i=1}^n \frac{q}{n} + (1 - q) \frac{1}{n_j} \sum_{i=1}^n A_{ji}$$

Ο πρώτος όρος $\frac{q}{n}$ είναι σταθερός και δεν εξαρτάται από το i . Αθροίζοντας τον n φορές, έχουμε:

$$\sum_{i=1}^n \frac{q}{n} = n \cdot \frac{q}{n} = q$$

Το άθροισμα $\sum_{i=1}^n A_{ji}$ ισούται με το πλήθος των εξερχόμενων συνδέσμων από τον κόμβο j , δηλαδή εξ ορισμού ισούται με n_j . Επομένως:

$$(1 - q) \frac{1}{n_j} \sum_{i=1}^n A_{ji} = (1 - q) \frac{1}{n_j} \cdot n_j = (1 - q) \cdot 1 = 1 - q$$

Προσθέτοντας τα δύο μέρη καταλήγουμε στο:

$$\sum_{i=1}^n G_{ij} = q + (1 - q) = 1$$

Αποδείχθηκε ότι το άθροισμα κάθε στήλης του πίνακα G είναι ίσο με 1, άρα ο πίνακας G είναι στοχαστικός. Η παραπάνω λογική υλοποιήθηκε στη μέθοδο `create_gm()` της κλάσης `PageRankSolver` στο αρχείο `ask4_lib.py`, στην οποία υπολογίζονται οι βαθμοί εξόδου n_j με την εντολή `np.sum(self.A, axis = 1)` και ο πίνακας G γεμίζει επαναληπτικά εφαρμόζοντας τον τύπο $G_{ij} = \frac{q}{n} + (1 - q) \frac{A_{ji}}{n_j}$.

4.2 Έλεγχος

Για την υλοποίηση του ελέγχου χρησιμοποιήθηκε η συνάρτηση `create_gm()` που περιγράψαμε παραπάνω και η `power_method()` για την εύρεση του ιδιοδιανύσματος που αντιστοιχεί στη μέγιστη ιδιοτιμή με τη Μέθοδο των Δυνάμεων. Η συνάρτηση αυτή λειτουργεί ως εξής:

1. Το διάνυσμα κατάταξης p αρχικοποιείται με ομοιόμορφη κατανομή ($1/n$ για κάθε σελίδα).
2. Σε κάθε βήμα k , υπολογίζεται το νέο διάνυσμα μέσω του πολλαπλασιασμού πίνακα-διανύσματος: $p^{(k+1)} = G \cdot p^{(k)}$.
3. Η διαδικασία τερματίζεται όταν η διαφορά μεταξύ δύο διαδοχικών διανυσμάτων (μετρούμενη με την L_1 νόρμα) γίνει μικρότερη από την καθορισμένη ανοχή ($tol = 1e - 7$).

Εφαρμόζουμε τον αλγόριθμο στο γράφημα της άσκησης με $n = 15$ κόμβους και $q = 0.15$ (Τα περνάμε ως ορίσματα στη δημοθργία αντικειμένου *PRSolver*).

```
Σύγκλιση σε 39 επαναλήψεις.  
  
Υπολογισμένο PageRank:  
[0.026825 0.029861 0.029861 0.026825 0.039587 0.039587 0.039587 0.039587  
0.074564 0.10632 0.10632 0.074564 0.125092 0.116328 0.125092]  
  
Top 5 Σελίδες:  
Page 15: 0.125092  
Page 13: 0.125092  
Page 14: 0.116328  
Page 11: 0.106320  
Page 10: 0.106320
```

Σχήμα 14: Αποτελέσματα Pagerank στα δεδομένα της εκφώνησης

Συγκρίνοντας τα αποτελέσματα του κώδικα με το διάνυσμα που δίνεται στην εκφώνηση της εργασίας παρατηρούμε πλήρη ταύτιση των τιμών.

4.3 Βελτίωση Βαθμού Σημαντικότητας

Επιλέγουμε να βελτιώσουμε τον βαθμό σημαντικότητας της σελίδας 5. Βασιζόμενοι στη λειτουργία του αλγορίθμου PageRank, γνωρίζουμε ότι μια σελίδα αποκτά υψηλότερη βαθμολογία όταν δέχεται συνδέσμους από σελίδες που είναι ήδη σημαντικές. Για το λόγο αυτό θα ανεβάσουμε την 5 δημιουργώντας συνδέσμους από τις σελίδες 15, 13, 14 και 11 προς αυτή. Θα καταργήσουμε τη σύνδεση $5 \Rightarrow 1$. Οι αλλαγές αυτές γίνονται πολύ εύκολα αλλάζοντας τα 0 και 1 στον πίνακα γειτνίασης A . Δημιουργούμε ένα νέο αντικείμενο *PRSolver* με τον τροποποιημένο πίνακα A_mod και καλούμε την *power_method()*. Παίρνουμε τα παρακάτω αποτελέσματα:

```
PageRank Σελίδας 5 (Αρχικό):      0.039587
PageRank Σελίδας 5 (Βελτιωμένο):  0.147020
```

Σχήμα 15: Βαθμός Σημαντικότητας της σελίδας 5, πριν και μετά τις μετατροπές

Βλέπουμε ότι το Pagerank της σελίδας 5 αυξήθηκε δραματικά, αποδεικνύοντας ότι η στρατηγική προσέλευσης συνδέσμων από κόμβους υψηλής σημαντικότητας είναι εξαιρετικά αποτελεσματική.

4.4 Αλλαγές στην πιθανότητα μεταπήδησης q

Για να δούμε τι προκαλούν οι αλλαγές του q κάνουμε τα εξής:

- Χρησιμοποιούμε μια `for loop` για να τρέξουμε τον αλγόριθμο δύο φορές, μία με $q=0.02$ και μία με $q=0.6$
- Για κάθε τιμή, δημιουργούμε ένα νέο αντικείμενο `PRSolver` με τον τροποποιημένο πίνακα `A_mod` από το προηγούμενο ερώτημα και καλούμε την `power_method`.
- Τυπώνουμε τις 3 κορυφαίες σελίδες και το PageRank της Σελίδας 5, για να δούμε πώς επηρεάζεται η κατάταξη.

```
Για q = 0.02:  
Top 3 Pages: [13, 10, 5]  
PageRank Σελίδας 5: 0.167637  
  
Για q = 0.6:  
Top 3 Pages: [10, 5, 13]  
PageRank Σελίδας 5: 0.097772
```

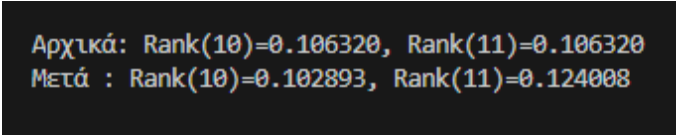
Σχήμα 16: Αποτελέσματα αλλαγών q

Το $q = 0.02$ δηλώνει πολύ μικρή πιθανότητα άλματος, δηλαδή ο χρήστης ακολουθεί σχεδόν πάντα τους συνδέσμους (98%) και σπάνια πηδάει τυχαία. Για $q = 0.6$ υπάρχει πολύ μεγάλη πιθανότητα άλματος, δηλαδή ο χρήστης πάει πιο εύκολα σε τυχαίες σελίδες (60% πιθανότητα). Η δομή των συνδέσμων χάνει τη δύναμή της. Το πλεονέκτημα που δώσαμε στη Σελίδα 5 μειώνεται από 0.167 σε 0.097, καθώς η επιρροή των συνδέσμων περιορίζεται από τον θόρυβο των τυχαίων αλμάτων. Από τα παραπάνω καταλαβαίνουμε ότι σκοπός της πιθανότητας μεταπήδησης είναι η μοντελοποίηση συμπεριφοράς, προσομοιώνοντας την πιθανότητα ένας χρήστης να σταματήσει να ακολουθεί συνδέσμους και να ανοίξει μια νέα σελίδα. Επιπλέον, επιτρέπει στον αλγόριθμο να βγει από αδιέξοδα ή κύκλους.

4.5 Τροποποίηση Βαρών

Για να δούμε αν η αλλαγή στον πίνακα γειτνίασης από 1 σε 3 επηρεάζει την κατάταξη ακολουθούμε τα εξής βήματα.

1. Ξεκινάμε πάλι από το αρχικό γράφημα της εκφώνησης για να μετρήσουμε την επίδραση μόνο της συγκεκριμένης αλλαγής.
2. Αντικαθιστούμε το $A_{(8,11)}$ και το $A_{(12,11)}$ με την τιμή 3 και τρέχουμε ξανά τη Μέθοδο των Δυνάμεων.
3. Υπολογίζουμε την κατάσταση πριν και μετά την αλλαγή.



Αρχικά: Rank(10)=0.106320, Rank(11)=0.106320
Μετά : Rank(10)=0.102893, Rank(11)=0.124008

Σχήμα 17: Αποτελέσματα Τροποποίησης Βαρών

Η στρατηγική αύξησης των βαρών πέτυχε, καθώς η σελίδα 11 ξεπέρασε την 10. Αυτό συμβαίνει διότι, αυξάνοντας το βάρος των συνδέσεων, η σελίδα 11 απορροφά μεγαλύτερο ποσοστό της πιθανότητας επίσκεψης από τις σελίδες 8 και 12, εις βάρος των άλλων σελίδων που συνδέονται με αυτές.

4.6 Επίδραση Διαγραφής Σελίδας

Θα μελετήσουμε την επίδραση της διαγραφής της σελίδας 10 στο συνολικό δίκτυο, ακολουθώντας τα παρακάτω βήματα:

1. Διαγραφή Γραμμής/Στήλης:

Ξεκινάμε από ένα νέο αντίγραφο του πίνακα A της εκφώνησης για να μην επηρεαστούν οι δοκιμές από τις αλλαγές των προηγούμενων ερωτημάτων. Η Σελίδα 10 αντιστοιχεί στον δείκτη 9. Χρησιμοποιώντας την εντολή *np.delete*, αφαιρούμε την 9η γραμμή και την 9η στήλη. Ο νέος πίνακας έχει διάσταση 14×14 .

2. Επαναυπολογισμός και Ενημέρωση Δεικτών:

Λύνουμε το νέο σύστημα με τη Μέθοδο των Δυνάμεων και ενημερώνουμε τους δείκτες του πίνακα (οι δείκτες των σελίδων μετά τη 10 μετατοπίστηκαν κατά μία θέση αριστερά).

3. Σύγκριση:

Συγκρίνουμε τα νέα ranks και υπολογίζουμε την ποσοστιαία διαφορά για κάθε σελίδα. Η ενημέρωση των δεικτών, ο υπολογισμός των διαφορών και η εκτύπωση γίνονται όλα σε ένα for-loop. Τα αποτελέσματα της εκτέλεσης παρουσιάζονται στον παρακάτω πίνακα.

Σελίδα	Πρίν	Μετά	Διαφορά
1	0.026825	0.047095	+75.57%
2	0.029861	0.040911	+37.01%
3	0.029861	0.035936	+20.34%
4	0.026825	0.032070	+19.55%
5	0.039587	0.042801	+8.12%
6	0.039587	0.041391	+4.56%
7	0.039587	0.051659	+30.49%
8	0.039587	0.050249	+26.93%
9	0.074564	0.048223	-35.33%
11	0.106320	0.170963	+60.80%
12	0.074564	0.103598	+38.94%
13	0.125092	0.041162	-67.09%
14	0.116328	0.107462	-7.62%
15	0.125092	0.186480	+49.07%

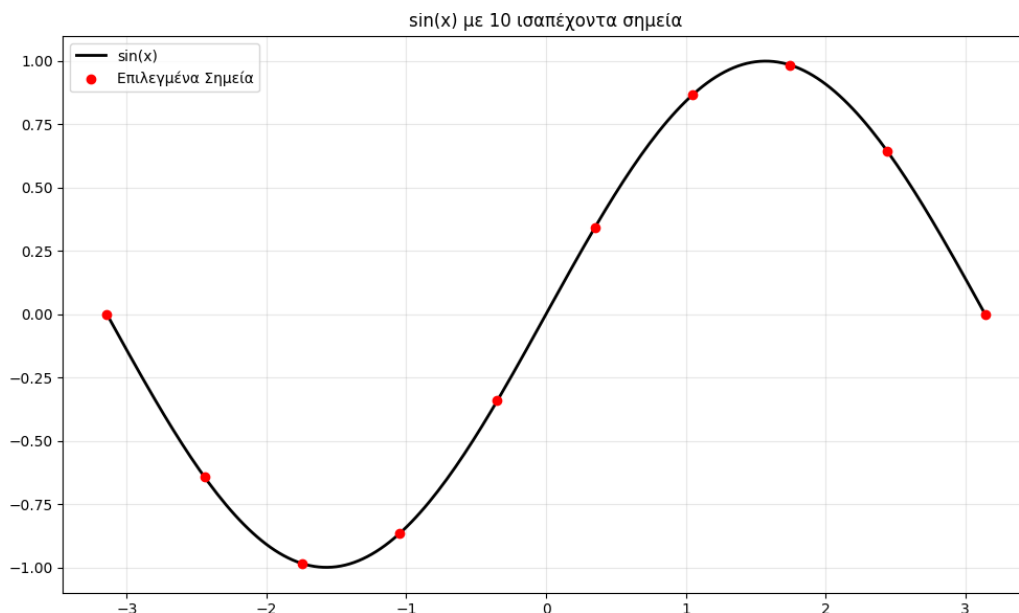
Σχήμα 18: Αποτελέσματα της διαγραφής της σελίδας 10

Ερμηνεία και Ανάλυση

Πριν τη διαγραφή, η σελίδα 10 δεχόταν συνδέσμους από τις σελίδες 5, 6, 7, 9, 14 και διοχέτευε όλη τη σημαντικότητά της αποκλειστικά στη σελίδα 13. Διαγράφοντας την 10, η 13 έχασε την τροφοδοσία της από έναν κόμβο υψηλής σημασίας. Αυτό εξηγεί τη δραματική πτώση (-67%). Η Σελίδα 9 δέχεται συνδέσμους από τις σελίδες 1 και 13. Παρόλο που η 1 ανέβηκε, η 13 κατέρρευσε παρασύροντας και την 9 προς τα κάτω (-35%). Οι σελίδες 6, 7 και 14 τροφοδοτούσαν προηγούμενως και την 10 όσο την 11. Με τη διαγραφή της 10, ο βαθμός εξόδου αυτών των σελίδων μειώθηκε. Αυτό σημαίνει ότι η σημαντικότητα του PageRank που μοιράζονται οι 6, 7 και 14 μοιράζεται σε λιγότερους κόμβους. Έτσι, η Σελίδα 11 παίρνει το μερίδιο που χάθηκε από την 10 ($+60\%$). Παρομοίως, η σελίδα 5 συνέδεε την 1 και την 10. Με την αφαίρεση της 10, η 5 διοχετεύει πλέον όλη τη σημαντικότητά της αποκλειστικά στη σελίδα 1, αυξάνοντας τη δύναμή της ($+75\%$). Η Σελίδα 15 τροφοδοτείται κυρίως από τη Σελίδα 11, η οποία έχει μοναδικό εξερχόμενο σύνδεσμο προς την 15. Επειδή η Σελίδα 11 σημείωσε τεράστια αύξηση, ανέβασε και τη 15 ($+49\%$). Επίσης και η 14 δεν δίνει πλέον στην 10, άρα το μερίδιο σημαντικότητας που δίνει στην 15 αυξάνεται.

Συμπέρασμα: Η διαγραφή ενός κόμβου λειτουργεί ευεργετικά για τις σελίδες που μοιράζονταν τους ίδιους γονείς, όπως οι 1 και 11 και καταστροφικά για αυτές που τροφοδοτούνταν από αυτόν, όπως η 13.

5 Προσέγγιση Τιμής Ημιτόνου



Σχήμα 1: Επιλογή σημείων στην $\sin(x)$

Αρχικά επιλέγουμε τα 10 σημεία τα οποία θα χρησιμοποιηθούν ως δεδομένα από τις συναρτήσεις. Χρησιμοποιούμε την `linspace` της `numpy` για να πάρουμε 10 σημεία με ίση απόσταση μεταξύ τους, τα οποία αποθηκεύουμε σε πίνακες (`x_train` και `y_train`).

5.1 Πολυωνυμική Προσέγγιση (Lagrange)

Η μέθοδος `lagrange` στο `ask5_lib.py` δέχεται τα σημεία `x_points`, `y_points` (δεδομένα) και το σημείο `x_val` στο οποίο θέλουμε να εκτιμήσουμε την τιμή του ημιτόνου. Ο κώδικας χρησιμοποιεί δύο εμφωλευμένους βρόχους (`loops`). Η συνάρτηση υλοποιεί τον κλασικό τύπο της παρεμβολής Lagrange:

$$P(x) = \sum_{i=0}^{n-1} y_i L_i(x)$$

όπου

$$L_i(x) = \frac{(x - x_0) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)}$$

Για κάθε σημείο i (εξωτερικός βρόχος), υπολογίζουμε τον όρο $L_i(x)$ ο οποίος κατασκευάζεται στον εσωτερικό βρόχο (for j) πολλαπλασιάζοντας τους όρους $\frac{x_{val}-x_j}{x_i-x_j}$ για όλα τα $j \neq i$. Έτσι, $L_i(x_j) = 1$ αν $i = j$ και 0 αν $i \neq j$. Τέλος, η συνάρτηση αθροίζει τα γινόμενα $y_i \cdot L_i(x)$ για να παράγει την τελική τιμή παρεμβολής.

5.2 Splines

Για τις Splines, υλοποιήσαμε τον αλγόριθμο εύρεσης των συντελεστών a_i, b_i, c_i, d_i για κάθε υποδιάστημα $[x_i, x_{i+1}]$, ώστε το κυβικό πολυώνυμο να έχει τη μορφή:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Η διαδικασία υλοποιείται στη συνάρτηση `calc_spl`:

1. Θέτουμε $a_i = y_i$ για όλα τα $i = 0, \dots, n - 1$.
2. Κατασκευάζουμε το γραμμικό σύστημα $A \cdot c_{in} = r$ για τους εσωτερικούς κόμβους ($i = 1, \dots, n - 1$). Τα στοιχεία του A και του r εξαρτώνται από τα βήματα h_i και τις τιμές y_i . Στον κώδικα, επιλύουμε αυτό το σύστημα για να βρούμε τα c_i στα εσωτερικά σημεία, θέτοντας $c_0 = 0$ και $c_n = 0$.
3. Υπολογίζονται αναλυτικά από τις τιμές των c_i και y_i με βάση τους τύπους:

$$b_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i(c_{i+1} + 2c_i)}{3}, \quad d_i = \frac{c_{i+1} - c_i}{3h_i}$$

Αφού βρεθούν τα c_i , οι υπόλοιποι συντελεστές προκύπτουν επιλύοντας τις εξισώσεις συνέχειας της spline και της δεύτερης παραγώγου της. Το d_i υπολογίζεται από τη γραμμική μεταβολή της δεύτερης παραγώγου στο διάστημα και το b_i υπολογίζεται λύνοντας την εξίσωση $S_i(x_{i+1}) = y_{i+1}$ ως προς b_i .

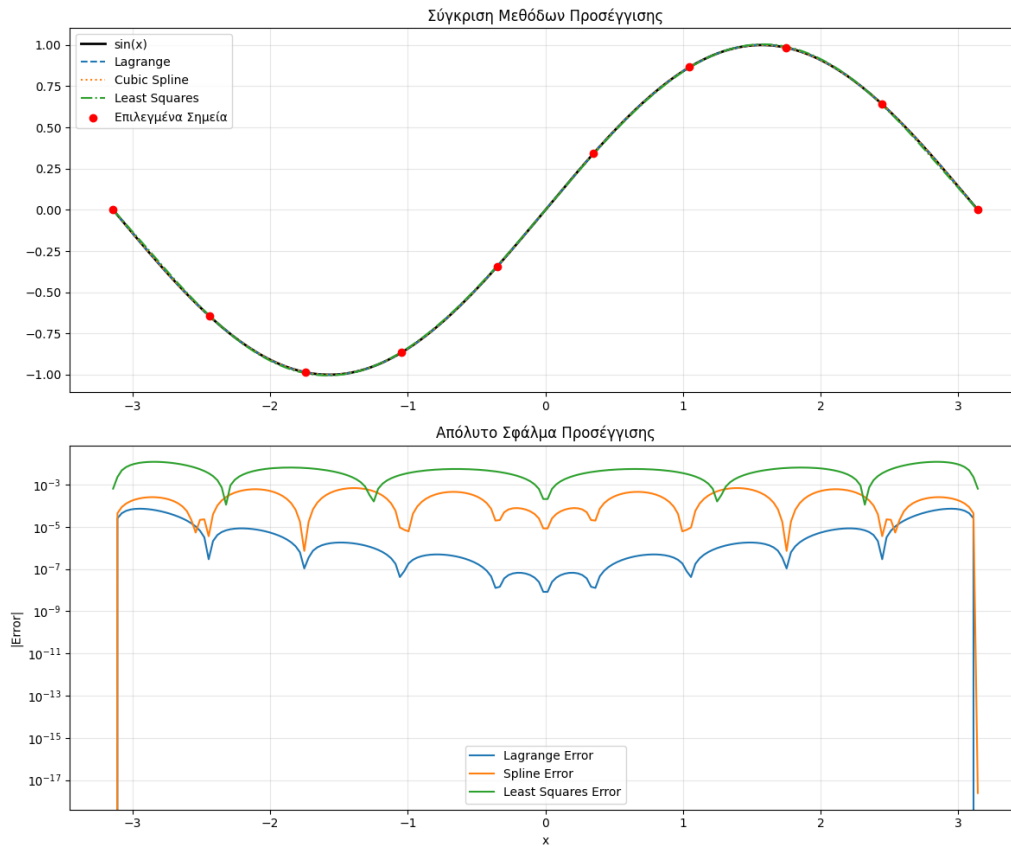
Η συνάρτηση `eval_spl` δέχεται τους υπολογισμένους συντελεστές και μια τιμή x . Αρχικά, εντοπίζει σε ποιο διάστημα $[x_i, x_{i+1}]$ ανήκει το x και στη συνέχεια εφαρμόζει τον τύπο του πολυωνύμου $S_i(x)$ χρησιμοποιώντας τα αντίστοιχα a_i, b_i, c_i, d_i .

5.3 Μέθοδος Ελάχιστων Τετραγώνων (Least Squares)

Η συνάρτηση `least_squares` δέχεται τα σημεία `x_points`, `y_points` (δεδομένα) και το σημείο `x_val` στο οποίο θέλουμε να εκτιμήσουμε την τιμή του ημιτόνου. Προσαρμόζει ένα πολυώνυμο βαθμού $deg = 5$ στα $n = 10$ σημεία. Ένα πολυώνυμο υψηλού βαθμού (π.χ. 8 ή 9) θα προσπαθούσε να ταιριάζει "τέλεια" στα 10 σημεία, αλλά θα εμφάνιζε έντονες ταλαντώσεις (κοιλίες και κορυφές) ανάμεσα στα σημεία, ειδικά στα άκρα του διαστήματος. Το πολυώνυμο 5ου βαθμού είναι αρκετό για να δώσει μια ομαλή καμπύλη που μοιάζει με το ημίτονο, αποφεύγοντας αυτές τις ταλαντώσεις. Τα βήματα:

1. Κατασκευάζουμε τον πίνακα σχεδιασμού A διαστάσεων $n \times (deg + 1)$, όπου η j -οστή στήλη περιέχει τις δυνάμεις x^j (Η 1η στήλη είναι όλο 1, η 2η είναι τα x , η 3η τα x^2 κτλ.).
2. Κατασκευάζουμε το σύστημα $A^T A c = A^T b$, όπου c οι άγνωστοι συντελεστές και b οι τιμές y .
3. Λύνουμε το γραμμικό σύστημα για την εύρεση των βέλτιστων συντελεστών.
4. Υπολογίζουμε την τιμή του πολυωνύμου στο `x_val` ($P(x) = c_0 + c_1 * x + c_2 * x^2 + \dots$)

5.4 Αποτελέσματα



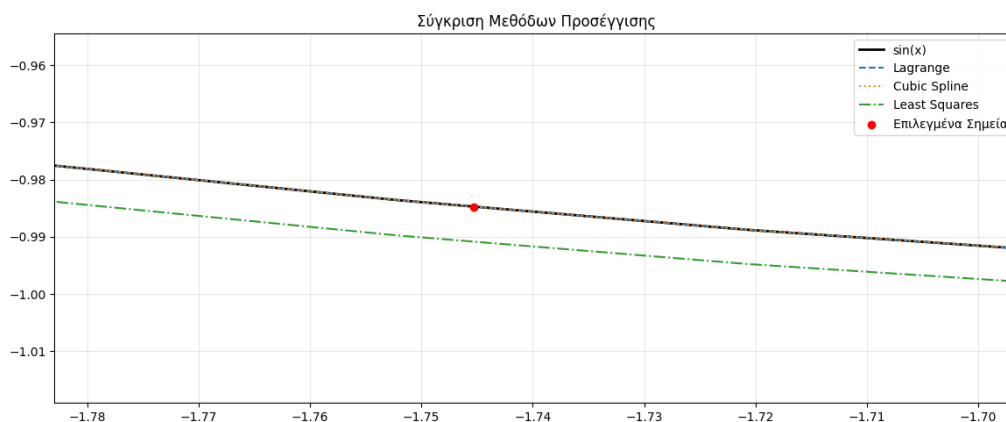
Σχήμα 2: Αποτελέσματα ask5.py

Στο πάνω γράφημα βλέπουμε ότι και οι τρεις μέθοδοι προσεγγίζουν την καμπύλη του ημιτόνου με μεγάλη επιτυχία, καθώς οι γραμμές αλληλοκαλύπτονται οπτικά. Τα κόκκινα σημεία υποδεικνύουν τα 10 σημεία εκπαίδευσης που χρησιμοποιήθηκαν. Στο κάτω γράφημα παρουσιάζεται η απόλυτη διαφορά $|f(x) - P(x)|$ σε λογαριθμική κλίμακα. Η μέθοδος Lagrange έδωσε το μικρότερο σφάλμα ($\approx 10^{-5}$) για το συγκεκριμένο πρόβλημα. Οι κυβικές Splines έδωσαν σφάλμα ελαφρώς μεγαλύτερο ($\approx 10^{-3}$). Η αξία των Splines δεν φαίνεται τόσο στην απόλυτη ακρίβεια σε απλές συναρτήσεις όπως το $\sin(x)$, αλλά στη σταθερότητά τους. Αν αυξάναμε τα σημεία, το Lagrange πιθανότατα θα αποτύγχανε στα άκρα, ενώ οι Splines θα συνέχιζαν να βελτιώνονται. Η μέθοδος Ελαχίστων Τετραγώνων με πο-

λυώνυμο 5ου βαθμού πέτυχε ακρίβεια συγκρίσιμη με τις Splines. Αυτό αποδεικνύει ότι ένα πολυώνυμο βαθμού 5 αρκεί για να μοντελοποιήσει την καμπύλη του ημιτόνου στο διάστημα $[-\pi, \pi]$.

Συμπέρασμα: Με τη μέθοδο Lagrange πετυχαίνουμε ακρίβεια περίπου 5 δεκαδικών ψηφίων (σφάλμα τάξης 10^{-5}), ενώ με τις Splines και τα Ελάχιστα Τετράγωνα πετυχαίνουμε ακρίβεια 3 δεκαδικών ψηφίων.

Αν μεγεθύνουμε τις γραφικές παραστάσεις μπορούμε να δούμε και στο γράφημα, ποια μέθοδος πλησίασε καλύτερα τη $\sin(x)$.

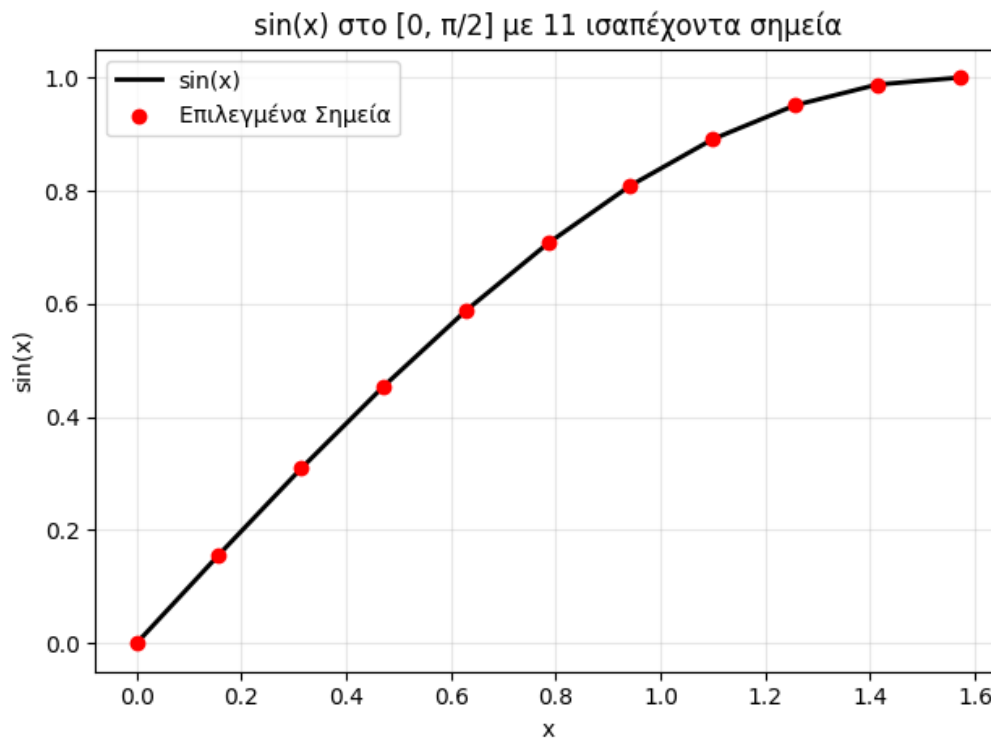


Σχήμα 3: Μεγέθυνση γραφήματος

Αυτό είναι αναμενόμενο, καθώς η μέθοδος ελάχιστων τετραγώνων είναι μέθοδος προσέγγισης και όχι παρεμβολής. Δεν προσπαθεί να περάσει από τα σημεία, αλλά να ελαχιστοποιήσει το συνολικό σφάλμα γύρω από αυτά.

6 Ολοκλήρωμα Ημιτόνου στο $[0, \frac{\pi}{2}]$

Θα υπολογίσουμε το ολοκλήρωμα $\int_0^{\pi/2} \sin(x) dx$ χρησιμοποιώντας 11 σημεία ($N = 10$ υποδιαστήματα) με τις μεθόδους του Τραπεζίου και του Simpson.



Σχήμα 4: $\sin(x)$ στο $[0, \frac{\pi}{2}]$

Επιλέγουμε 11 ισαπέχοντα σημεία πάνω στη $\sin(x)$ ($x = \text{np.linspace}(a, b, \text{simeia})$) και φτιάχνουμε έναν πίνακα y με τις τιμές της συνάρτησης στα σημεία αυτά ($y = f(x)$).

6.1 Μέθοδος Τραπεζίου

Εφαρμόζουμε τον κανόνα του τραπεζίου:

$$I \approx \frac{b-a}{2N} \left[y_0 + 2 \sum_{i=1}^{N-1} y_i + y_N \right]$$

Στον κώδικα, αυτό υλοποιήθηκε αθροίζοντας τα ενδιάμεσα στοιχεία του πίνακα ($y[1:-1]$) πολλαπλασιασμένα επί 2, και προσθέτοντας τα άκρα ($y[0]$, $y[-1]$).

6.2 Μέθοδος Simpson

Η μέθοδος του Simpson απαιτεί άρτιο αριθμό υποδιαστημάτων. Στην περίπτωσή μας έχουμε 11 σημεία \rightarrow 10 διαστήματα, επομένως μπορούμε να προχωρήσουμε. Εφαρμόζουμε τον τύπο:

$$I \approx \frac{b-a}{3N} \left[y_0 + 4 \sum_{i=1}^{\frac{N}{2}} y_{2i-1} + 2 \sum_{i=1}^{\frac{N}{2}-1} y_{2i} + y_N \right]$$

Στον κώδικα, χρησιμοποιήθηκε η τεχνική του slicing για τον διαχωρισμό των ενδιάμεσων σημείων. $y[1:-1:2]$, σημεία με περιττό δείκτη (y_1, y_3, \dots) που πολλαπλασιάζονται με 4 και $y[2:-1:2]$, σημεία με άρτιο δείκτη (y_2, y_4, \dots), που πολλαπλασιάζονται με 2.

6.3 Υπολογισμός Σφαλμάτων

Το αριθμητικό σφάλμα είναι η απόλυτη διαφορά μεταξύ της προσεγγιστικής τιμής και της πραγματικής ($I_{true} = 1.0$). Το θεωρητικό σφάλμα υπολογίζεται από του εξής τύπους:

- Τραπεζίο: $|E_T| \leq \frac{(b-a)^3}{12N^2} \max |f''(x)|$
- Simpson: $|E_S| \leq \frac{(b-a)^5}{180N^4} \max |f^{(4)}(x)|$

Για την $\sin(x)$ στο $[0, \pi/2]$, η μέγιστη τιμή της 2ης και 4ης παραγώγου είναι 1.

6.4 Αποτελέσματα και Συμπεράσματα

```
Μέθοδος Τραπεζίου:  
Αποτέλεσμα: 0.99794299  
Αριθμητικό Σφάλμα: 0.00205701  
Θεωρητικό Φράγμα: 0.00322982  
  
Μέθοδος Simpson:  
Αποτέλεσμα: 1.00000339  
Αριθμητικό Σφάλμα: 0.00000339  
Θεωρητικό Φράγμα: 0.00000531
```

Σχήμα 5: Αποτελέσματα εκτέλεσης ask6.py

Η μέθοδος Simpson πέτυχε εξαιρετικά υψηλή ακρίβεια (σφάλμα τάξης 10^{-6}), ενώ η μέθοδος του Τραπεζίου είχε σφάλμα τάξης 10^{-3} . Αυτό συμβαίνει διότι η μέθοδος Simpson προσεγγίζει τη συνάρτηση τμηματικά με παραβολές, οι οποίες προσαρμόζονται πολύ καλύτερα στην καμπυλότητα της συνάρτησης $\sin(x)$ σε σχέση με τις ευθείες γραμμές (τραπέζια). Και στις δύο περιπτώσεις, το πραγματικό αριθμητικό σφάλμα που υπολογίσαμε είναι μικρότερο από το θεωρητικό σφάλμα που δίνει ο τύπος.

7 Εκτίμηση τιμών του Χρηματιστηρίου Αθηνών

Για τα δεδομένα, παίρνουμε τις τιμές κλεισίματος των 10 συνεδριάσεων πριν την 28/05 (από 14/05/2025 έως 27/05/2025), από πραγματικά δεδομένα του Χρηματιστηρίου στο www.capital.gr, για τις εταιρείες AKTOR(AKTR) και TITAN(TITC).

27/5/2025 ▼	5,3200	27/5/2025 ▲	41,6500
26/5/2025 ▲	5,3400	26/5/2025 ▲	40,5500
23/5/2025 ▼	5,2900	23/5/2025 ▼	39,9000
22/5/2025 ▼	5,3500	22/5/2025 ▼	40,7000
21/5/2025 ▼	5,3800	21/5/2025 ▼	41,5500
20/5/2025 ▲	5,3900	20/5/2025 ▲	41,9500
19/5/2025 ▼	5,3500	19/5/2025 ▼	41,5500
16/5/2025 ▲	5,3900	16/5/2025 ▲	41,8000
15/5/2025 ▼	5,3700	15/5/2025 ▼	41,5500
14/5/2025 ▲	5,3800	14/5/2025 ▼	41,7500

Σχήμα 6: Τιμές κλεισίματος AKTOR (αριστερά) και TITAN (δεξιά)

Οι πραγματικές τιμές κλεισίματος των μετοχών για τις 28/05/2025 είναι 5.34 και 41.85 αντίστοιχα.

7.1 Ανάλυση Κώδικα

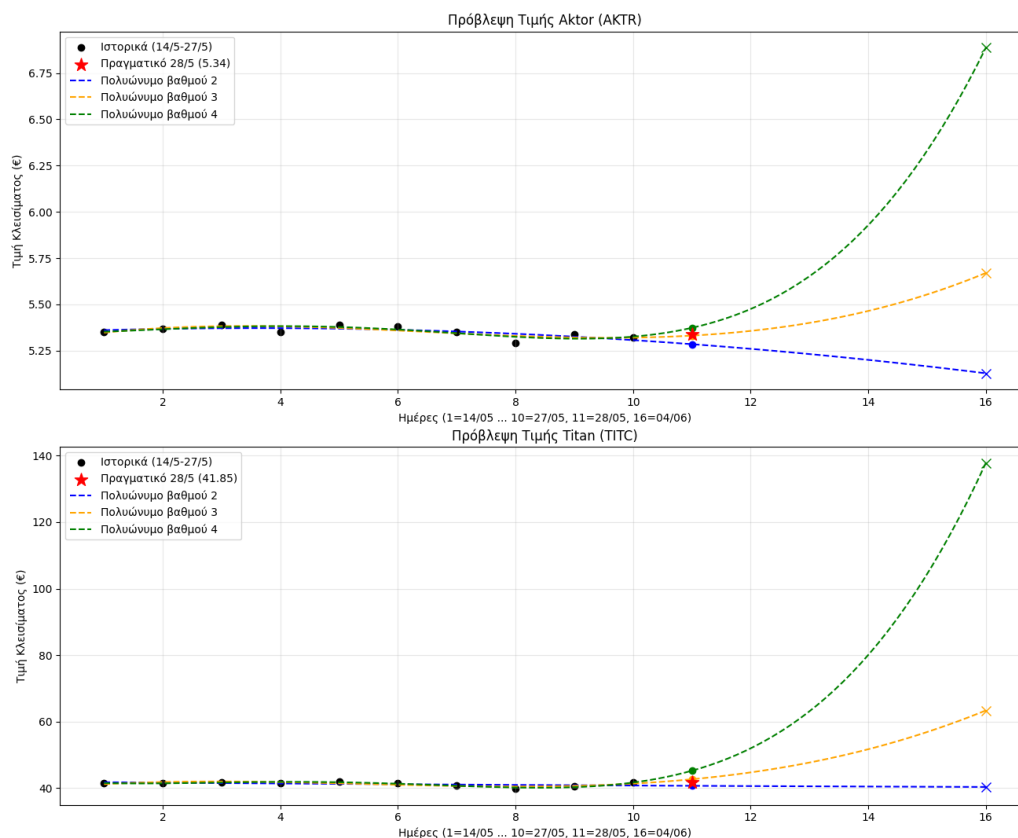
Για τον αλγόριθμο πρόβλεψης χρησιμοποιούμε την συνάρτηση *least_squares* που αναλύσαμε στην Άσκηση 5. Στην κύρια ροή, αντιστοιχίζουμε τις ημερομηνίες των 10 συνεδριάσεων στους ακεραίους $x \in \{1, 2, \dots, 10\}$. Το πρόγραμμα εκτελείται για κάθε μετοχή ξεχωριστά. Για κάθε βαθμό πολωνύμου ($m = 2, 3, 4$) δημιουργούμε ένα πυκνό σύνολο σημείων (`np.linspace`) στο διάστημα $[1, 16]$. Για κάθε σημείο, καλούμε τη *least_squares* ώστε να προκύψει η ομαλή καμπύλη προσρμογής και για τα σημεία $x = 11$ (28/05/2025) και $x = 16$ (04/06/2025) για τις προβλέψεις.

7.2 Αποτελέσματα και Σχολιασμός

```
--- Ανάλυση για Aktor (AKTR) ---  
Πραγματική τιμή 28/5: 5.34  
[Βαθμός 2]  
  Πρόβλεψη 28/5: 5.2850 (Απόκλιση: 0.0550)  
  Πρόβλεψη 04/06: 5.1277  
[Βαθμός 3]  
  Πρόβλεψη 28/5: 5.3313 (Απόκλιση: 0.0087)  
  Πρόβλεψη 04/06: 5.6698  
[Βαθμός 4]  
  Πρόβλεψη 28/5: 5.3733 (Απόκλιση: 0.0333)  
  Πρόβλεψη 04/06: 6.8866  
  
--- Ανάλυση για Titan (TITC) ---  
Πραγματική τιμή 28/5: 41.85  
[Βαθμός 2]  
  Πρόβλεψη 28/5: 40.7208 (Απόκλιση: 1.1292)  
  Πρόβλεψη 04/06: 40.3693  
[Βαθμός 3]  
  Πρόβλεψη 28/5: 42.6867 (Απόκλιση: 0.8367)  
  Πρόβλεψη 04/06: 63.3682  
[Βαθμός 4]  
  Πρόβλεψη 28/5: 45.2542 (Απόκλιση: 3.4042)  
  Πρόβλεψη 04/06: 137.7539
```

Σχήμα 7: Αποτελέσματα εκτέλεσης ask7.py

Η μετοχή της AKTOR κινήθηκε σε μικρό εύρος (5.29€ - 5.39€) με μικρές διακυμάνσεις. Η πραγματική τιμή στις 28/5 είναι 5.34€. Το πολυώνυμο 3ου βαθμού πέτυχε την καλύτερη προσέγγιση με πρόβλεψη 5.3313€ (απόκλιση μόλις 0.0087€). Το πολυώνυμο 2ου βαθμού υποτίμησε την τιμή (5.2850€), ενώ του 4ου βαθμού άρχισε να υπερεκτιμά (5.3733€). Για τις 04/06 βλέπουμε έντονη απόκλιση στις προβλέψεις. Ενώ ο 2ος βαθμός δείχνει πτώση στα 5.12€, ο 4ος βαθμός προβλέπει εκρηκτική άνοδο στα 6.88€. Δεδομένης της ιστορικής σταθερότητας της μετοχής, η πρόβλεψη του 4ου βαθμού είναι υπερπροσαρμογή.



Σχήμα 8: Διαγράμματα Προβλέψεων

Η μετοχή της TITAN παρουσίασε έντονη πτώση και απότομη ανάκαμψη. Η πραγματική τιμή στις 28/5 είναι 41.85€. Το πολυώνυμο 3ου βαθμού ήταν το πλησιέστερο με πρόβλεψη 42.68€ (απόκλιση 0.83€). Στην πρόβλεψη για τις 04/06 φαίνεται και πάλι το πρόβλημα των πολυωνύμων υψηλού βαθμού. Το πολυώνυμο 4ου βαθμού προβλέπει τιμή 137.75€ για τις 4 Ιουνίου και το πολυώνυμο 3ου βαθμού προβλέπει 63.36€. Και οι δύο τιμές είναι εξωπραγματικές για διάστημα 5 ημερών (υπονοούν υπερδιπλασιασμό της αξίας). Αυτό συμβαίνει διότι τα πολυώνυμα υψηλού βαθμού τείνουν να ταλαντώνονται έντονα στα άκρα για να ταιριάζουν στα δεδομένα εκπαίδευσης, καθιστώντας τα ακατάλληλα για μακροπρόθεσμες προβλέψεις εκτός δείγματος.

Συμπεραίνουμε ότι η αύξηση του βαθμού του πολυωνύμου μειώνει το σφάλμα στα ιστορικά δεδομένα αλλά αυξάνει δραματικά τον κίνδυνο λανθασμένων προβλέψεων στο μέλλον.