ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΑΘΗΜΑ: ΑΡΙΘΜΗΤΙΚΗ ΑΝΑΛΥΣΗ

1 ΥΠΟΧΡΕΩΤΙΚΗ ΕΡΓΑΣΙΑ

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: ΔΗΜΗΤΡΙΟΣ ΠΑΝΤΕΛΕΗΜΩΝ ΓΙΑΚΑΤΟΣ

ΑΕΜ: 3061

ΠΕΡΙΕΧΟΜΕΝΑ

Άσκηση 1

Στην παραπάνω άσκηση μας δίνεται η συνάρτηση $f(x) = 14xe^{x-2} - 12e^{x-2} - 7x^3 + 20x^2 - 26x + 12$ στο διάστημα [0, 3]. Σκοπός μας είναι να δημιουργήσουμε την γραφική παράσταση της συνάρτησης σε αυτό το διάστημα. Να υπολογίσουμε με πρόγραμμα τις ρίζες της με ακρίβεια 6 δεκαδικά ψηφία χρησιμοποιώντας:

- Τη μέθοδο διχοτόμησης.
- Τη μέθοδο Newton – Raphson.
- Τη μέθοδο τέμνουσας.

Επίσης, μας ζητείτε να συγκρίνουμε το πλήθος των επαναλήψεων που έγιναν, να αποδείξουμε για ποιες ρίζες η μέθοδο Newton – Raphson συγκλίνει τετραγωνικά και για ποιες όχι και να αιτιολογήσουμε ποιο είναι το χαρακτηριστικό των ριζών για τις οποίες η προηγούμενη μέθοδο που αναφέραμε δεν συγκλίνει τετραγωνικά.

Αρχικά χρησιμοποιώντας το πρόγραμμα Matlab δημιουργούμε την γραφική παράσταση της δοθείσας συνάρτησης και βλέπουμε από το γράφημα ότι η συγκεκριμένη συνάρτηση έχει κάποιες ρίζες στο διάστημα [0, 3]. Το επόμενο βήμα είναι να σπάσουμε το διάστημα [0, 3] σε υπό διαστήματα και με τις προαναφερόμενες τρεις μεθόδους να υπολογίζουμε τις ρίζες της συνάρτησης στα διαστήματα αυτά:

- Μέθοδο διχοτόμησης: Αρχικά υπολογίζουμε το πλήθος των επαναλήψεων που απαιτούνται ώστε να έχουμε την επιθυμητή ακρίβεια των έξι δεκαδικών ψηφίων. Μετά, μέσα σε μία δομή επανάληψης for βρίσκουμε κάθε φορά την ενδεχόμενη ρίζα υπολογίζοντας το μέσο του διαστήματος που επιλέγουμε κάθε φορά και στη συνέχεια εξετάζουμε αν αυτή η νέα τιμή (ενδεχόμενη ρίζα) που βρίσκουμε κάθε φορά σε ένα καινούργιο διάστημα είναι μία ρίζα της συνάρτησης που εξετάζουμε. Αν είναι μία ρίζα τότε σταματάει η επαναληπτική δομή και το πρόγραμμα μας επιστρέφει τα ζητούμενα αποτελέσματα.
- Μέθοδο Newton – Raphson: Αρχικά υπολογίζουμε την πρώτη και δεύτερη παράγωγο της δοθείσας συνάρτησης και μετά επιλέγουμε με μία δομή επιλογής if την αρχική ρίζα στο διάστημα που εξετάζουμε. Ύστερα μέσα σε μία δομή επανάληψης while υπολογίζουμε την νέα ρίζα κάθε φορά και αμέσως μετά με μία δομή επιλογής εξετάζουμε αν η απόλυτη διαφορά την νέας ρίζας με την προηγούμενη είναι μικρότερη από το σφάλμα. Αν αυτή η διαφορά είναι μικρότερη από το σφάλμα τότε σταματάει η δομή επανάληψης και το πρόγραμμα μας επιστρέφει τα ζητούμενα αποτελέσματα. Αν η διαφορά δεν είναι τότε η επαναληπτική δομή συνεχίζεται μέχρι να βρεθεί η ρίζα.
- Μέθοδο τέμνουσας: Αρχικά με μία δομή επιλογής if εξετάζουμε αν υπάρχει μία μοναδική ρίζα στο διάστημα που έχουμε επιλέξει. Αν δεν υπάρχει τότε το πρόγραμμα τερματίζει καθώς δεν ισχύον οι προϋποθέσεις της συγκεκριμένης μεθόδου. Αν υπάρχει ρίζα τότε επιλέγουμε ως δύο ρίζες τα δύο άκρα του διαστήματος και μετά με μία δομή επανάληψης while εξετάζουμε αν η διαφορά σε απόλυτη τιμή ων δύο ριζών είναι μεγαλύτερη από το σφάλμα. Αν η συνθήκη είναι αληθής τότε συνεχίζει να εκτελείται η επαναληπτική δομή υπολογίζοντας κάθε φορά την νέα ρίζα. Αν δεν ισχύει η συνθήκη τότε το πρόγραμμα τερματίζει επιστέφοντας τα ζητούμενα αποτελέσματα.

Στη συνέχεια, θα παρουσιάσουμε τις ρίζες της συνάρτησης καθώς και πόσες επαναλήψεις χρειαστήκανε οι παραπάνω μέθοδοι για να βρεθούν οι ρίζες:

- Μέθοδο διχοτόμησης:
    - Διάστημα [0, 1.5]:
        - Ρίζα: 0.8571428060531616
        - Επαναλήψεις: 22
    - Διάστημα [1.5, 3]:
        - Ρίζα: 2.0000038146972656
        - Επαναλήψεις: 17
- Μέθοδο Newton – Raphson:
    - Διάστημα [0, 1.5]:
        - Ρίζα: 0.8571428571428563
        - Επαναλήψεις: 7
    - Διάστημα [1.5, 3]:
        - Ρίζα: 1.9999875369187812
        - Επαναλήψεις: 43
- Μέθοδο τέμνουσας:
    - Διάστημα [0, 1]:
        - Ρίζα: 0.8571428571428537
        - Επαναλήψεις: 8
    - Διάστημα [1.5, 3]:
        - Ρίζα: 2.0000023163198097
        - Επαναλήψεις: 40

Από τα παραπάνω αποτελέσματα βλέπουμε ότι σε κάποια διαστήματα οι μέθοδοι Newton – Raphson και τέμνουσας είναι πιο γρήγοροι σε σχέση με την μέθοδο της διχοτόμησης ενώ σε κάποια άλλα διαστήματα γίνεται το αντίστροφο.

Σε αυτό το σημείο θα δείξουμε για ποιες ρίζες η μέθοδο Newton – Raphson συγκλίνει τετραγωνικά και για ποιες όχι:

Από το ανάπτυγμα Taylor της δοθείσας συνάρτησης με κέντρο το σημείο x^* το οποίο είναι η ρίζα της στης συνάρτησης έχουμε:

$$f(x_n) = f(x^*) + f'(x^*)(x_n - x^*) + \frac{f''(\xi_n)}{2}(x_n - x^*)^2, \xi_n \in (x_n, x^*) \text{ ή } \xi_n \in (x^*, x_n)$$

$$f'(x_n) = f'(x^*) + f''(\xi'_n)(x_n - x^*), \xi_n \in (x_n, x^*) \text{ ή } \xi_n \in (x^*, x_n)$$

Αντικαθιστούμε τις τιμές $f(x_n), f'(x_n)$ στον αναδρομικό τύπο $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$, γνωρίζουμε ότι $f(x^*) = 0$, επειδή το σημείο x^* είναι ρίζα της συνάρτησης και παίρνουμε:

$$x_{n+1} = x_n - \frac{f(x^*) + f'(x^*)(x_n - x^*) + \frac{f''(\xi_n)}{2}(x_n - x^*)^2}{f'(x^*) + f''(\xi'_n)(x_n - x^*)}$$

$$x_{n+1} = x_n - \frac{f'(x^*)(x_n - x^*) + \frac{f''(\xi_n)}{2}(x_n - x^*)^2}{f'(x^*) + f''(\xi'_n)(x_n - x^*)}$$

$$x_{n+1} - x^* = x_n - x^* - \frac{f'(x^*)(x_n - x^*) + \frac{f''(\xi_n)}{2}(x_n - x^*)^2}{f'(x^*) + f''(\xi'_n)(x_n - x^*)}$$

$$x_{n+1} - x^* = \frac{(x_n - x^*)f'(x^*) + f''(\xi'_n)(x_n - x^*)^2 - f'(x^*)(x_n - x^*) - \frac{f''(\xi_n)}{2}(x_n - x^*)^2}{f'(x^*) + f''(\xi'_n)(x_n - x^*)}$$

$$x_{n+1} - x^* = \frac{f''(\xi'_n)(x_n - x^*)^2 - \frac{f''(\xi_n)}{2}(x_n - x^*)^2}{f'(x^*) + f''(\xi'_n)(x_n - x^*)}$$

$$x_{n+1} - x^* = (x_n - x^*)^2 \frac{f''(\xi'_n) - \frac{f''(\xi_n)}{2}}{f'(x^*) + f''(\xi'_n)(x_n - x^*)}$$

$$\frac{x_{n+1} - x^*}{(x_n - x^*)^2} = \frac{f''(\xi'_n) - \frac{f''(\xi_n)}{2}}{f'(x^*) + f''(\xi'_n)(x_n - x^*)}$$

$$\lim_{n\to\infty} \frac{x_{n+1} - x^*}{(x_n - x^*)^2} = \lim_{n\to\infty} \frac{f''(\xi'_n) - \frac{f''(\xi_n)}{2}}{f'(x^*) + f''(\xi'_n)(x_n - x^*)}$$

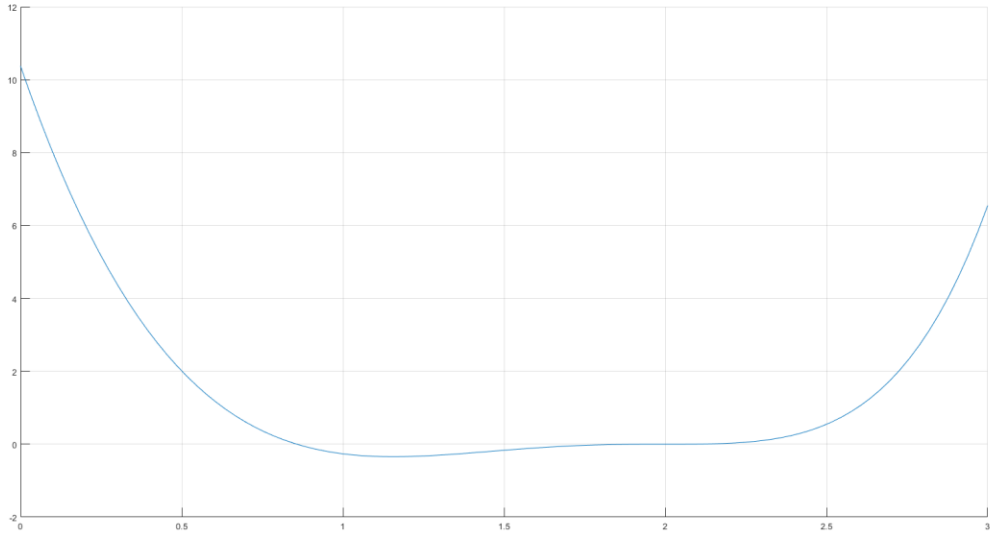
$$\lim_{n\to\infty} \frac{x_{n+1} - x^*}{(x_n - x^*)^2} = \frac{f''(x^*)}{2f'(x^*)}$$

Επομένως για να συγκλείνει τετραγωνικά το $\lim_{n\to\infty} \frac{x_{n+1}-x^*}{(x_n-x^*)^2} = \frac{f''(x^*)}{2f'(x^*)} \neq 0$.

Έπειτα θα εξετάσουμε αν οι ρίζες που βρήκαμε με τη μέθοδο Newton – Raphson συγκλίνουν τετραγωνικά ή δεν συγκλίνουν:

- Για $x^* = 0.8571428571428563$: $\lim_{n\to\infty} \frac{x_{n+1}-x^*}{(x_n-x^*)^2} = \frac{f''(x^*)}{2f'(x^*)} = -2.413850894567762 \neq 0$. Άρα η ρίζα συγκλίνει τετραγωνικά.
- Για $x^* = 2.0000038146972656 \cong 2$: $\lim_{n\to\infty} \frac{x_{n+1}-x^*}{(x_n-x^*)^2} = \frac{f''(x^*)}{2f'(x^*)} = 0$. Άρα η ρίζα δεν συγκλίνει τετραγωνικά.

Τέλος, το κύριο χαρακτηριστικό των ριζών για τις οποίες η μέθοδο Newton – Raphson δεν συγκλίνει τετραγωνικά είναι ότι αν σε ένα διάστημα υπάρχουν πάρα πολλές ρίζες έτσι ώστε η γραφική παράσταση σε εκείνο το διάστημα να είναι ευθεία, τότε η σύγκληση δεν είναι τετραγωνική αλλά γραμμική.



Εικόνα 1: Γραφική παράσταση της συνάρτησης f.

Υπόδειξη: Στον φάκελο Άσκηση 1 υπάρχουν αρχεία που περιλαμβάνουν τον κώδικα της γραφικής παράστασης σε Matlab και των μεθόδων σε Python.

Άσκηση 2

Στην συγκεκριμένη άσκηση μας ζητείτε να υλοποιήσουμε μία τροποποιημένη μέθοδο Newton – Raphson, μία τροποποιημένη μέθοδο διχοτόμησης και μία τροποποιημένη μέθοδο τέμνουσας ώστε να βρούμε όλες τις ρίζες της συνάρτησης $f(x) = 94cos^3x - 24cosx + 177sin^2x - 108sin^4x - 72cos^3x\ sin^2x - 65$ στο διάστημα [0, 3] με ακρίβεια έξι δεκαδικών ψηφίων. Επίσης, σκοπός μας είναι να εκτελέσουμε τον δεύτερο τροποποιημένο αλγόριθμο (μέθοδο διχοτόμησης) δέκα φορές για να διαπιστώσουμε αν συγκλίνει πάντα σε ίδιο αριθμό επαναλήψεων και να συγκρίνουμε τις τροποποιημένες μεθόδους σε σχέση με τις κλασικές ως προς την ταχύτητα σύγκλισης.

Αρχικά υλοποιούμε τις μεθόδους:

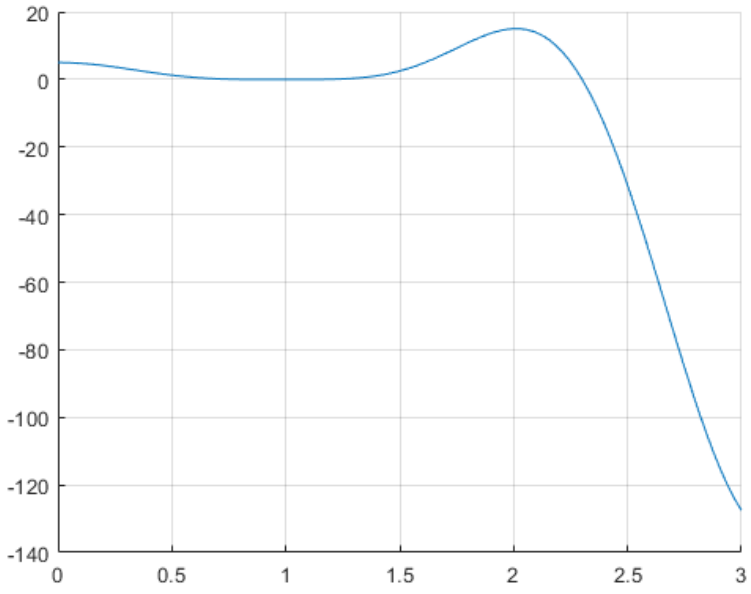
- Τροποποιημένη μέθοδο Newton – Raphson: Αρχικά υπολογίζουμε την πρώτη και δεύτερη παράγωγο της δοθείσας συνάρτησης και μετά επιλέγουμε με μία δομή επιλογής if την αρχική ρίζα στο διάστημα που εξετάζουμε. Ύστερα μέσα σε μία δομή επανάληψης while υπολογίζουμε την νέα ρίζα κάθε φορά με βάση την τροποποιημένη μορφή και αμέσως μετά με μία δομή επιλογής εξετάζουμε αν η απόλυτη διαφορά την νέας ρίζας με την προηγούμενη είναι μικρότερη από το σφάλμα. Αν αυτή η διαφορά είναι μικρότερη από το σφάλμα τότε σταματάει η δομή επανάληψης και το πρόγραμμα μας επιστρέφει τα ζητούμενα αποτελέσματα. Αν η διαφορά δεν είναι τότε η επαναληπτική δομή συνεχίζεται μέχρι να βρεθεί η ρίζα.
- Τροποποιημένη μέθοδο διχοτόμησης: Αρχικά υπολογίζουμε την ενδεχόμενη ρίζα με βάση την τροποποιημένη μορφή. Μετά, μέσα σε μία δομή επανάληψης while βρίσκουμε κάθε φορά σε ένα καινούργιο διάστημα μία ενδεχόμενη ρίζα της συνάρτησης. Εξετάζουμε αν αυτή η νέα τιμή (ενδεχόμενη ρίζα) που βρίσκουμε κάθε φορά σε ένα καινούργιο διάστημα είναι μία ρίζα της συνάρτησης. Αν είναι μία ρίζα τότε σταματάει η επαναληπτική δομή και το πρόγραμμα μας επιστρέφει τα ζητούμενα αποτελέσματα. Αν δεν είναι, τότε αμέσως μετά με μία δομή επιλογής εξετάζουμε αν η απόλυτη διαφορά την νέας ρίζας με την προηγούμενη είναι μικρότερη από το σφάλμα. Αν αυτή η διαφορά είναι μικρότερη από το σφάλμα τότε σταματάει η δομή επανάληψης και το πρόγραμμα μας επιστρέφει τα ζητούμενα αποτελέσματα. Αν η διαφορά δεν είναι τότε η επαναληπτική δομή συνεχίζεται μέχρι να βρεθεί η ρίζα.
- Τροποποιημένη μέθοδο τέμνουσας: Αρχικά με μία δομή επιλογής if εξετάζουμε αν υπάρχει μία μοναδική ρίζα στο διάστημα που έχουμε επιλέξει. Αν δεν υπάρχει τότε το πρόγραμμα τερματίζει καθώς δεν ισχύον οι προϋποθέσεις της συγκεκριμένης μεθόδου. Αν υπάρχει ρίζα τότε επιλέγουμε ως δύο ρίζες, δύο από τα τρία σημεία του διαστήματος και μετά με μία δομή επανάληψης while εξετάζουμε αν η διαφορά σε απόλυτη τιμή ων δύο ριζών είναι μεγαλύτερη από το σφάλμα. Αν η συνθήκη είναι αληθής τότε συνεχίζει να εκτελείται η επαναληπτική δομή υπολογίζοντας κάθε φορά την νέα ρίζα με βάση την τροποποιημένη μορφή. Αν δεν ισχύει η συνθήκη τότε το πρόγραμμα τερματίζει επιστέφοντας τα ζητούμενα αποτελέσματα.

Στη συνέχεια, θα παρουσιάσουμε τις ρίζες της συνάρτησης καθώς και πόσες επαναλήψεις χρειαστήκανε οι παραπάνω μέθοδοι για να βρεθούν οι ρίζες:

- Τροποποιημένη μέθοδο Newton – Raphson:
  - Διάστημα [0, 1.5]:
    - Ρίζα: 1.0471945658769912
    - Επαναλήψεις: 22
  - Διάστημα [0, 2.5]:

        - Ρίζα: 2.300523983021863
        - Επαναλήψεις: 4
- Τροποποιημένη μέθοδο διχοτόμησης:
    - Διάστημα [0, 2.5]:
        - Ρίζα: 2.3005239105678794
        - Επαναλήψεις: 32
    - Διάστημα [0, 3]:
        - Ρίζα: 0.8410705885860714
        - Επαναλήψεις: 29
- Τροποποιημένη μέθοδο τέμνουσας:
    - Διάστημα [0, 0.5] και [0.5, 1.5]:
        - Ρίζα: 1.0290885919183639
        - Επαναλήψεις: 29

Ύστερα, διαπιστώνουμε ότι η τροποποιημένη μέθοδο διχοτόμησης δεν συγκλείνει πάντα στον ίδιο αριθμό αφού, εκτελέσουμε τον αλγόριθμο για δέκα φορές. Αυτό συμβαίνει επειδή κάθε φορά επιλέγεται ένας τυχαίος αριθμός για ρίζα στο διάστημα που εξετάζουμε και άρα σε κάθε εκτέλεση θα εμφανίζεται και μία διαφορετική ρίζα. Τέλος, αν συγκρίνουμε τις τροποποιημένες μεθόδους σε σχέση με τις κλασικές ως προς την ταχύτητα συμπεραίνουμε ότι οι κλασικές μέθοδοι είναι σχετικά πιο γρήγορες από τις τροποποιημένες και αυτό φαίνεται από τον αριθμό των επαναλήψεων στις σχεδόν ίδιες ρίζες (με βάση τη γραφική παράσταση καταλήγουμε ότι η συνάρτηση που εξετάζουμε έχει πάρα πολλές ρίζες στο διάστημα [0, 3] και λόγο floating τα δεκαδικά ψηφία των ριζών θα είναι διαφορετικά σε κάθε μέθοδο) ότι στις κλασικές είναι πιο μικρός από τις τροποποιημένες.



Εικόνα 2: Γραφική παράσταση της συνάρτησης f.

Υπόδειξη: Στον φάκελο Άσκηση 2 υπάρχουν αρχεία που περιλαμβάνουν τον κώδικα της γραφικής παράστασης σε Matlab και των μεθόδων σε Python.

Άσκηση 3

Στο πρώτο της συγκεκριμένης άσκησης μας ζητείτε να προγραμματίσουμε μία συνάρτηση που θα επιλύει το γραμμικό σύστημα $Ax = b$ και θα δέχεται σαν είσοδο τον πίνακα A και το διάνυσμα b και θα επιστρέφει σαν έξοδο το διάνυσμα των αγνώστων x. Το πρόγραμμα περιλαμβάνει πέντε συναρτήσεις οι οποίες υλοποιούν την μέθοδο επίλυσης γραμμικών συστημάτων $PA = LU$. Παρακάτω θα αναλύσουμε την συγκεκριμένη υλοποίηση.

Αρχικά πολλαπλασιάζουμε το γραμμικό σύστημα $Ax = b$ με τον πίνακα P, που περιλαμβάνει την πληροφορία για τις γραμμές που ανταλλάχθηκαν μεταξύ τους κατά την διαδικασία του Gauss με οδήγηση στον πίνακα A.

$$PAx = Pb$$

Στη συνέχεια, επειδή $PA = LU$:

$$LUx = Pb$$

Μετά:

$$Lz = Pb$$

$$Ux = z$$

Οι πίνακες L και U, δηλαδή ο κάτω και ο άνω τριγωνικός πίνακας αντίστοιχα, προκύπτουν μετά την διαδικασία του Gauss με οδήγηση, οι οποίοι αποτελούν ένα πολύ χρήσιμο εργαλείο για την γρήγορη επίλυση του γραμμικού συστήματος.

Σε αυτό το σημείο θα αναλύσουμε τις πέντε συναρτήσεις που κατασκευάστηκαν για την υλοποίηση της παραπάνω διαδικασίας:

1. solverU: Η συνάρτηση δέχεται ως είσοδο ένα άνω τριγωνικό πίνακα U και ένα διάνυσμα z και επιστρέφει ως έξοδο το διάνυσμα των αγνώστων x, δηλαδή υπολογίζεται η σχέση $Ux = z$ από την παραπάνω ανάλυση. Η επίλυση του γραμμικού συστήματος ξεκινάει από την τελευταία γραμμή του πίνακα U και του πίνακα b και τελειώνει στην πρώτη γραμμή των αντίστοιχων πινάκων.
2. solverL: Η συνάρτηση δέχεται ως είσοδο ένα κάτω τριγωνικό πίνακα L και ένα διάνυσμα Pb και επιστρέφει ως έξοδο το διάνυσμα των αγνώστων z, δηλαδή υπολογίζεται η σχέση $Lz = Pb$ από την παραπάνω ανάλυση. Μετά την εύρεση του διανύσματος z, το συγκεκριμένο διάνυσμα χρησιμοποιείται ως είσοδο στην συνάρτηση solverU. Η επίλυση του γραμμικού συστήματος ξεκινάει από την πρώτη γραμμή του πίνακα L και του πίνακα Pb και τελειώνει στην τελευταία γραμμή των αντίστοιχων πινάκων.
3. multi: Η συνάρτηση δέχεται ως είσοδο ένα πίνακα και ένα διάνυσμα και επιστρέφει ως έξοδο το διάνυσμα που προκύπτει μετά τον πολλαπλασιασμό. Αρχικά η συνάρτηση ελέγχει αν πληρούνται οι συνθήκες για τον πολλαπλασιασμό του πίνακα με το διάνυσμα, δηλαδή αν ο αριθμός των στηλών του πίνακα είναι ίσος με τον αριθμό των γραμμών του διανύσματος. Αν δεν ισχύει η συνθήκη τότε το πρόγραμμα τερματίζει διαφορετικά πολλαπλασιάζει κάθε γραμμή του πίνακα με την στήλη του διανύσματος.
4. PLU: Η συνάρτηση δέχεται ως είσοδο ένα πίνακα A και επιστρέφει ως έξοδο τους τρεις πίνακες P, L και U. Στην συγκεκριμένη συνάρτηση υλοποιείται ο αλγόριθμος του Gauss με οδήγηση και

αυτό γιατί, ο συγκεκριμένος αλγόριθμος αποτελεί το κύριο κορμό για την εύρεση των τριών πινάκων που θέλουμε να επιστρέψουμε ως έξοδο. Πιο αναλυτικά, αρχικά χρησιμοποιούμε μία εμφωλευμένη δομή επανάληψης for για να βρούμε το μέγιστο κατά απόλυτη τιμή στοιχείο στην στήλη που εξετάζουμε κάθε φορά. Αφού βρούμε αυτό το στοιχείο τότε κάνουμε ανταλλαγή μεταξύ των γραμμών που περιέχει το συγκεκριμένο στοιχείο και της γραμμής που θα χρησιμοποιηθεί ως η οδηγός γραμμή. Την πληροφορία για το ποιες γραμμές ανταλλάχθηκαν μεταξύ του την αποθηκεύουμε στον πίνακα P. Σε αυτό το σημείο είναι απαραίτητο να τονιστεί ότι η οδηγός γραμμή σε κάθε επανάληψη θα είναι η αμέσως επόμενη γραμμή. Ύστερα, βρίσκουμε το οδηγό στοιχείο (το πρώτο μη μηδενικό στοιχείο) σε κάθε οδηγό γραμμή και μηδενίζουμε όλα τα στοιχεία που βρίσκονται κάτω από το οδηγό στοιχείο. Μετά, τοποθετούμε το οδηγό στοιχείο στην αντίστοιχη θέση του κάτω τριγωνικού πίνακα L. Επιπρόσθετα τα στοιχεία της κύρια διαγώνιου του πίνακα L είναι παντού 1. Τέλος, μετά την ολοκλήρωση του Gauss με οδήγηση ο πίνακας που προέκυψε είναι ο άνω τριγωνικός πίνακας U.

5. finalSolver: Η συνάρτηση δέχεται ως είσοδο τον πίνακα A και το διάνυσμα b και επιστρέφει ως έξοδο το διάνυσμα των αγνώστων x. Πιο αναλυτικά η συγκεκριμένη συνάρτηση χρησιμοποιεί τις προαναφερόμενες συναρτήσεις για να βρει το τελικό διάνυσμα.

Στο δεύτερο μέρος της άσκηση μας ζητείτε να προγραμματίσουμε μία συνάρτηση η οποία θα παίρνει ως είσοδο έναν συμμετρικό και θετικά ορισμένο πίνακα και θα επιστρέφει ως έξοδο έναν κάτω τριγωνικό πίνακα που αποτελεί την αποσύνθεση Cholesky του πίνακα εισόδου. Πιο αναλυτικά το πρόγραμμα περιλαμβάνει τρεις συναρτήσεις:

1. AT: Η συνάρτηση δέχεται ως είσοδο ένα πίνακα και επιστρέφει τον αντίστροφο πίνακα του πίνακα εισόδου.
2. equal: Η συνάρτηση δέχεται ως είσοδο δύο πίνακας και ελέγχει αν είναι ίσοι. Αν οι πίνακες δεν είναι ίσοι τότε το πρόγραμμα τερματίζει, αφού παραβιάζει τον ορισμό του θεωρήματος Cholesky, δηλαδή ότι ο πίνακας που πρόκειται να γίνει ανάλυση Cholesky δεν είναι συμμετρικός.
3. cholesky: Η συνάρτηση δέχεται ως είσοδο ένα συμμετρικό πίνακα A και επιστρέφει ως έξοδο ένα κάτω τριγωνικό πίνακα L που αποτελεί την ανάλυση Cholesky. Αρχικά η συνάρτηση ελέγχει αν ο πίνακας εισόδου είναι συμμετρικός χρησιμοποιώντας τις δύο παραπάνω συναρτήσεις. Στη συνέχεια χρησιμοποιούμε μία εμφωλευμένη δομή επανάληψης for για τον σχηματισμό του πίνακα εξόδου. Τέλος, ο κάτω τριγωνικός πίνακας L θα σχηματιστεί από τις σχέσεις:
    a. $L_{i,j} = \frac{1}{L_{j,j}}\left(A_{i,j} + \sum_{k-1}^{j-1} L_{i,k}L_{j,k}\right), i > j$, όπου i γραμμή και j στήλη.
    b. $L_{i,i} = \sqrt{A_{i,i} + \sum_{k=1}^{i-1} L_{i,k}^2}\,, i = j$, όπου i γραμμή και j στήλη.

Στο τρίτο και τελευταίο μέρος της άσκησης μας ζητείται να προγραμματίσουμε την μέθοδο Gauss – Seidel και να την χρησιμοποιήσουμε για να επιλύσουμε με ακρίβεια τεσσάρων δεκαδικών ψηφίων το $n \times n$ αραιό σύστημα $Ax = b$ για $n = 10$ και για $n = 10000$, με $A(i,i) = 5$, $A(i,i+1) = -2$ και $b = [3, 1, 1, \dots, 1, 1, 3]^T$. Πιο αναλυτικά το πρόγραμμα περιλαμβάνει τρεις συναρτήσεις:

1. genA: Η συνάρτηση δέχεται ως είσοδο μία μεταβλητή n και επιστρέφει ως έξοδο τον πίνακα $A^{n\times n}$, n γραμμών και n στηλών, δηλαδή, η συνάρτηση δημιουργεί τον πίνακα A με $A(i,i) = 5$ και $A(i,i+1) = -2$.
2. genb: Η συνάρτηση δέχεται ως είσοδο μία μεταβλητή n και επιστρέφει ως έξοδο τον διάνυσμα b, δηλαδή, η συνάρτηση δημιουργεί το διάνυσμα $b = [3, 1, 1, \dots, 1, 1, 3]^T$.

3. GaussSeidel: Η συνάρτηση δέχεται ως είσοδο ένα πίνακα A και ένα διάνυσμα b και επιστρέφει ως έξοδο το διάνυσμα x και τη διαφορά των δύο τελευταίων διανυσμάτων ως προς την άπειρη νόρμα m. Αρχικά η συνάρτηση βρίσκει αν ο πίνακας εισόδου των συντελεστών των αγνώστων ενός γραμμικού συστήματος έχει κυριαρχική διαγώνιο, δηλαδή αν $|a_{i,i}| \geq \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{i,j}|, i,j = 1, \dots, n$ τότε η μέθοδος Gauss – Seidel συγκλίνει. Αν συγκλίνει τότε το αρχικό διάνυσμα που θα χρησιμοποιήσει ο αλγόριθμος θα θεωρηθεί αυθαίρετα ότι είναι 0, διαφορετικά θα θεωρηθεί ότι είναι 1. Στην συνέχεια χρησιμοποιώντας μία εμφωλευμένη δομή επανάληψης υπολογίζουμε κάθε φορά το νέο διάνυσμα με βάση την αναδρομική ακολουθία $x_i^{(m+1)} = \frac{1}{a_{i,i}}\left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(m+1)} - \sum_{j=i+1}^{n} a_{i,j} x_j^{(m)}\right), i = 1, \dots, n$ και $m = 1, \dots$. Τέλος, η επαναληπτική δομή θα σταματήσει μόνο όταν η διαφορά των δύο τελευταίων διανυσμάτων ως προς την άπειρη νόρμα είναι μικρότερη από την ακρίβεια τεσσάρων δεκαδικών ψηφίων.

Υπόδειξη: Στον φάκελο Άσκηση 3 υπάρχουν αρχεία που περιλαμβάνουν τον κώδικα των προγραμμάτων σε Python.

Άσκηση 4

Στην συγκεκριμένη άσκηση μας δίνετε ο πίνακας γειτνίασης $A \in \mathbb{R}^{n\times n}, n = 15$ του γραφήματος που απεικονίζεται στην εκφώνηση της άσκησης 4 στην 1 υποχρεωτική εργασία. Αρχικά, σκοπός μας είναι να κατασκευάσουμε τον πίνακα G, που το κελί $G_{(i,j)} = \frac{q}{n} + \frac{A_{(j,i)}(1-q)}{n_j}$, να αποδείξουμε ότι ο συγκεκριμένος πίνακας είναι στοχαστικός, δηλαδή το άθροισμα κάθε στήλης είναι 1, να ελέγξουμε ότι το ιδιοδιάνυσμα της μέγιστης ιδιοτιμής είναι αυτό που δίνει η εκφώνηση.

Σε αυτό το σημείο θα αποδείξουμε ότι ο πίνακας G είναι στοχαστικός:

Για την 1 στήλη έχουμε:

$$G_{(1,1)} + G_{(2,1)} + G_{(3,1)} + \cdots + G_{(n,1)} =$$

$$\frac{q}{n} + \frac{A_{(1,1)}(1-q)}{n_1} + \frac{q}{n} + \frac{A_{(1,2)}(1-q)}{n_1} + \frac{q}{n} + \frac{A_{(1,3)}(1-q)}{n_1} + \cdots + \frac{q}{n} + \frac{A_{(1,n)}(1-q)}{n_1} =$$

$$\left(\frac{q}{n} + \frac{q}{n} + \frac{q}{n} + \cdots + \frac{q}{n}\right) + \left(\frac{A_{(1,1)}(1-q)}{n_1} + \frac{A_{(1,2)}(1-q)}{n_1} + \frac{A_{(1,3)}(1-q)}{n_1} + \cdots + \frac{A_{(1,n)}(1-q)}{n_1}\right) =$$

$$n * \frac{q}{n} + \frac{(1-q)}{n_1} * \left(A_{(1,1)} + A_{(1,2)} + A_{(1,3)} + \cdots + A_{(1,n)}\right) =$$

$$q + \frac{(1-q)}{n_1} * n_1 =$$

$$q + 1 - q =$$

$$1$$

Γνωρίζουμε ότι: $\left(A_{(1,1)} + A_{(1,2)} + A_{(1,3)} + \cdots + A_{(1,n)}\right) = n_1$, που n_1είναι το πλήθος των συνδέσεων στην πρώτη στήλη.

Αντίστοιχα η ίδια διαδικασία και για τις υπόλοιπες $n-1$ στήλες. Άρα αφού όλες οι στήλες του πίνακα G έχουν άθροισμα 1 δείξαμε ότι ο συγκεκριμένος πίνακας είναι στοχαστικός.

Στην συνέχεια φτιάχνουμε ένα πρόγραμμα που θα μας βοηθήσει να κατασκευάσουμε τον πίνακα G και να ελέγξουμε ότι το ιδιοδιάνυσμα της μέγιστης ιδιοτιμής είναι αυτό που δίνει η εκφώνηση. Πιο αναλυτικά το πρόγραμμα έχει πέντε συναρτήσεις:

1. genG: Η συνάρτηση δέχεται ως είσοδο τον πίνακα A και την πιθανότητα μετακίνησης ενός χρήστη σε κάποια άλλη σελίδα q και επιστρέφει ως έξοδο τον πίνακα G. Για την κατασκευή του πίνακα G χρησιμοποιεί μία εμφωλευμένη δομή επανάληψης και το κάθε κελί του πίνακα προκύπτει από την σχέση $G_{(i,j)} = \frac{q}{n} + \frac{A_{(j,i)}(1-q)}{n_j}$.
2. multi: Η συνάρτηση δέχεται ως είσοδο ένα πίνακα και ένα διάνυσμα και επιστρέφει ως έξοδο το διάνυσμα που προκύπτει μετά τον πολλαπλασιασμό. Αρχικά η συνάρτηση ελέγχει αν πληρούνται οι συνθήκες για τον πολλαπλασιασμό του πίνακα με το διάνυσμα, δηλαδή αν ο αριθμός των στηλών του πίνακα είναι ίσος με τον αριθμό των γραμμών του διανύσματος. Αν δεν ισχύει η

συνθήκη τότε το πρόγραμμα τερματίζει διαφορετικά πολλαπλασιάζει κάθε γραμμή του πίνακα με την στήλη του διανύσματος.

3. multiMatrix: Η συνάρτηση δέχεται ως είσοδο δύο πίνακες και επιστρέφει ως έξοδο τον πίνακα που προκύπτει μετά τον πολλαπλασιασμό. Αρχικά η συνάρτηση ελέγχει αν πληρούνται οι συνθήκες για τον πολλαπλασιασμό δύο πινάκων, δηλαδή αν ο αριθμός των στηλών του πρώτου πίνακα είναι ίσος με τον αριθμό των γραμμών του δεύτερου πίνακα. Αν δεν ισχύει η συνθήκη τότε το πρόγραμμα τερματίζει διαφορετικά πολλαπλασιάζει κάθε γραμμή του πίνακα με τις στήλη του άλλου πίνακα.
4. PowerMethod: Η συνάρτηση δέχεται ως είσοδο ένα πίνακα και επιστρέφει το ιδιοδιάνυσμα του πίνακα εισόδου. Με λίγα λόγια η συνάρτηση βρίσκει το ιδιοδιάνυσμα του πίνακα χρησιμοποιώντας την μέθοδο των δυνάμεων. Η συνάρτηση με μία δομή επιλογής εξετάζει αν είναι η πρώτη φορά που εκτελείτε η συνάρτηση για να επιλέξει σωστά το διάνυσμα που θα χρησιμοποιήσει για το πολλαπλασιασμό με τον πίνακα εισόδου. Ύστερα με μία δομή επανάληψης σχηματίζει το ιδιοδιάνυσμα και με μία δομή επιλογής εξετάζει αν η απόλυτη τιμή της διαφοράς των δύο τελευταίων ιδιοτιμών είναι μικρότερο από το σφάλμα ώστε να τερματίσει η επαναληπτική διαδικασία και να επιστρέψει τα ζητούμενα δεδομένα.
5. genP: Η συνάρτηση δέχεται ως είσοδο ένα διάνυσμα και επιστρέφει ως έξοδο την κανονικοποίηση του διανύσματος εισόδου.

Με τις παραπάνω συναρτήσεις δημιουργούμε το διάνυσμα p που προκύπτει μετά την κανονικοποίηση του ιδιοδιανύσματος του πίνακα G μετά την μέθοδο των δυνάμεων και άρα το διάνυσμα είναι:

$$p = [0.026824547469370425, 0.029861082785477867, 0.029861085894394103, 0.026824554451507797,$$
$$0.039587221879650386, 0.039587223178495515, 0.0395872303252960 8, 0.03958723162414121,$$
$$0.07456441129676307, 0.10631991333920492, 0.10631992876698798, 0.07456442912195776,$$
$$0.12509158167335585, 0.1163279558161836, 0.12509160237721334]^T$$

Αν συγκρίνουμε τα δύο ιδιοδιανύσματα, δηλαδή το παραπάνω με το ιδιοδιάνυσμα της εκφώνησης θα παρατηρήσουμε ότι ενώ τα πρώτα στοιχεία είναι ακριβώς ίδια, τα τελευταία στοιχεία των αριθμών είναι διαφορετικά μεταξύ τους και αυτό συμβαίνει εξαετίας του floating που γίνεται στον υπολογιστή.

Στην συνέχεια θα προσθέσουμε 4 συνδέσεις και θα αφαιρέσουμε 1 από αυτές που ήδη υπάρχουν στο γράφημα. Επιλέγουμε να προσθέσουμε τις συνδέσεις $A_{(1,3)}, A_{(1,4)}, A_{(1,5)}$ και $A_{(1,6)}$ και αφαιρούμε την σύνδεση $A_{(2,3)}$. Μετά την εκτέλεση του προγράμματος το νέο ιδιοδιάνυσμα που προκύπτει είναι το:

$$p = [0.027946963777919116, 0.021615606029606202, 0.02702275741743877, 0.03073786810439381,$$
$$0.0422281696786 1077, 0.04069799172950493, 0.04100954279533731, 0.039479364846231456,$$
$$0.06734961935723711, 0.10669166289717902, 0.10826394034038025, 0.07702206065097933,$$
$$0.12562455076535992, 0.11734892259077769, 0.12696097901904435]^T$$

Παρατηρούμε ότι ο βαθμός σημαντικότητας της σελίδας 1 βελτιώθηκε σε σχέση με πριν, καθώς από 0.026824547469370425 έγινε 0.027946963777919116 ενώ ο βαθμός σημαντικότητας της 2 σελίδας υποβαθμίστηκε, καθώς από 0.029861082785477867 έγινε 0.021615606029606202.

Ύστερα θα αλλάξουμε την πιθανότητα μεταπήδησης q και θα περιγράψουμε τις αλλαγές στην τάξη σελίδας:

- Για $q = 0.02$:

$$p = [0.017106331422813488, 0.014428870809710635, 0.014428870434833136, 0.017106330626565257,$$
$$0.03218980261079931, 0.03218980244337174, 0.03218980161670933, 0.032189801449281755,$$
$$0.08002972327733544, 0.10957602275874266, 0.10957602096840446, 0.0800297210779703,$$
$$0.1434985025105366, 0.14196189793108024, 0.14349850006184553]^T$$

Παρατηρούμε ότι η τάξη σελίδας των σελίδων που ο χρήστης δεν επισκέπτεται συχνά έχει υποβαθμιστεί, καθώς η πιθανότητα μεταπήδησης έχει μειωθεί, ενώ η τάξη σελίδας των σελίδων με τις περισσότερες προβολές έχει αυξηθεί για τον ίδιο λόγο. Αυτό πρακτικά σημαίνει ότι ένας χρήστης έχει πολύ μικρή πιθανότητα να μεταπηδήσει σε μία άλλη σελίδα από αυτές που επισκέπτεται συνήθως.

- Για $q = 0.6$:

$$p = [0.05133212367026154, 0.057999720415342605, 0.057999720415342605, 0.05133212367026154,$$
$$0.056660610958948275, 0.056660610958948275, 0.056660610958948275, 0.056660610958948275,$$
$$0.06695487031076086, 0.09026137462310695, 0.09026137462310695, 0.06695487031076088,$$
$$0.08344224337834066, 0.07337689136858165, 0.08344224337834066]^T$$

Παρατηρούμε ότι η τάξη σελίδας των σελίδων που ο χρήστης δεν επισκέπτεται συχνά έχει βελτιωθεί, καθώς η πιθανότητα μεταπήδησης έχει αυξηθεί, ενώ η τάξη σελίδας των σελίδων με τις περισσότερες προβολές έχει υποβαθμιστεί για τον ίδιο λόγο. Αυτό πρακτικά σημαίνει ότι ένας χρήστης δεν μένει μόνο στις σελίδες με τις περισσότερες προβολές αλλά μεταπηδάει σε στις υπόλοιπες σελίδες.

Μετά, θα δοκιμάσουμε μία νέα στρατηγική προκυμμένου να βελτιώσουμε την τάξη της σελίδας 11 σε σχέση με τον ανταγωνιστή της, την σελίδα 10. Για αυτό το λόγο μοντελοποιούμε αυτή την κατάσταση αντικαθιστώντας το $A_{(8,11)}$ και το $A_{(12,11)}$ με 3 στον πίνακα γειτνίασης. Το ιδιοδιάνυσμα που προκύπτει είναι:

$$p = [0.026550499444273564, 0.028371718217431553, 0.025015598899717603, 0.016416336488082484,$$
$$0.03894238141860393, 0.03799148047795101, 0.031145442463482484, 0.03019454152282957,$$
$$0.07377788957928214, 0.1028932390383971, 0.12400838904145929, 0.07709874484004489,$$

$$0.1235150738324551, 0.12261572235028016, 0.14146294238570925]^T$$

Παρατηρούμε η τάξη σελίδας στην σελίδα 11 είναι 0.12400838904145929 ενώ στην σελίδα 10 είναι 0.1028932390383971, επειδή $0.12400838904145929 > 0.1028932390383971$ βλέπουμε ότι αυτή η στρατηγική δουλεύει.

Τέλος, θα μελετήσουμε την επίδραση της διαγραφής της σελίδας 10 από το γράφημα. Το ιδιοδιάνυσμα που προκύπτει είναι:

$$p = [0.04709520324127086, 0.04091142766170118, 0.035935567405281706, 0.03206998876568841,$$

$$0.04280070992058539, 0.041390993255976005, 0.051658676093960736, 0.05024895942935135,$$

$$0.048223404163048565, 0.1709628464298777, 0.1035981405504351, 0.041161947933822544,$$

$$0.10746215582580326, 0.1864799793231974]^T$$

Παρατηρούμε ότι μετά την διαγραφή της σελίδας 10, η σελίδα 11 αποκτάει μεγαλύτερο βαθμό σημαντικότητας (μεγαλύτερη τάξη σελίδας), όπως και κάποιες άλλες σελίδες που πριν είχαν μικρότερη τάξη, τώρα έχουν μεγαλύτερη, ενώ η σελίδα 13 πλέον έχει μικρότερη τάξη. Αυτό συμβαίνει επειδή η σελίδα 10 ήταν ένας μεγάλος κόμβος από άλλες σελίδες. Μετά την διαγραφή του οι άλλες σελίδες αύξησαν την τάξη τους γιατί ο χρήστης προκυμμένου να επισκεφτεί κάποια σελίδα από αυτές που είχε η σελίδα 10, είναι αναγκαίο να μεταπηδήσει και στις υπόλοιπες σελίδες.

Υπόδειξη: Στον φάκελο Άσκηση 4 υπάρχει αρχείο που περιλαμβάνει τον κώδικα του προγράμματος σε Python.