

1η Υποχρεωτική Εργασία LaTeX

Ονοματεπώνυμο: Στέφανος Καραμπέρας
AEM: 2910

20 Δεκεμβρίου 2019

Άσκηση 1

Γενικές παρατηρήσεις

Το προγραμματιστικό μέρος της εργασίας έχει υλοποιηθεί με χρήση Πηκτων 3. Ο κώδικας που έχει συνταχθεί για την άσκηση 1 βρίσκεται εντός του φακέλου “Code/task-1” .

Οι ζητούμενοι αλγόριθμοι που εφαρμόζουν τις τεχνικές προσέγγισης ρίζας με τη μέθοδο της διχοτόμησης, τη μέθοδο Newton-Raphson και τη μέθοδο της τέμνουσας, έχουν υλοποιηθεί εντός του αρχείου “root_estimation_algorithms.py”. Επιπρόσθετα, εντός του αρχείου f_function.py έχουν δημιουργηθεί οι συναρτήσεις calculate_f(x), calculate_der_1_f(x) και calculate_der_2_f(x). Οι συναρτήσεις αυτές δέχονται ως όρισμα έναν αριθμό ξ και επιστρέφουν την ρίζα της $f(x)$, $f'(x)$ και $f''(x)$ αντίστοιχα, για την δοθείσα συνάρτηση $f(x)$ της εκφώνησης της άσκησης 1. Τέλος, με δεδομένο πως η ζητούμενη ακρίβεια ορίζεται στο 5ο δεκαδικό ψηφίο, η μεταβλητή t_err στο αρχείο “root_estimation_algorithms.py” αρχικοποιείται με την τιμή $t_err = 0.000005$ (ακρίβεια 5ου δεκαδικού ψηφίου με στρογγυλοποίηση).

Αλγόριθμος για τη μέθοδο Διχοτόμησης

Η συνάρτηση που υλοποιήθηκε για την εκτέλεση του αλγορίθμου που εφαρμόζει τη μέθοδο της διχοτόμησης είναι η “partitioning_root_estimation()”.

Η συνάρτηση αρχικά υπολογίζει το πλήθος των απαιτούμενων επαναλήψεων της μεθόδου διχοτόμησης για την επίτευξη της ζητούμενης ακρίβειας. Στη συνέχεια, ξεκινά να εκτελεί το ορισμένο πλήθος επαναλήψεων, σταματώντας σε περίπτωση που βρει ακριβή ρίζα της $f(x)$ και τυπώνοντας τη ρίζα, ή συνεχίζοντας μέχρι να ολοκληρώσει το ορισμένο πλήθος επαναλήψεων, τυπώνοντας τελικά την προσέγγιση της ρίζας και το πλήθος επαναλήψεων που απαιτήθηκαν για την εκτίμησή της.

Αλγόριθμος για τη μέθοδο Newton-Raphson

Η συνάρτηση που υλοποιήθηκε για την εκτέλεση του αλγορίθμου που εφαρμόζει τη μέθοδο Newton-Raphson είναι η `newton_raphson_root_estimation()`.

Η συνάρτηση αρχικοποιεί το x_{n-1} χρησιμοποιώντας το πρώτο στοιχείο του πεδίου ορισμού $[-2,2]$ της $f(x)$, δηλαδή το -2 . Ακολουθώντας, υπολογίζει διαδοχικά το x_n μέχρις ότου η απόλυτη τιμή της διαφοράς $x_{n-1} - x_n$ να είναι μικρότερη ή ίση του καθορισμένου αποδεκτού σφάλματος (ακρίβεια), δηλαδή $|x_n - x_{n-1}| \leq t_err$. Στο τέλος κάθε επανάληψης, η τιμή του q_{n-1} ανανεώνεται με την τιμή του q_n . Η συνάρτηση καταγράφει σε μια μεταβλητή `iter_count` το πλήθος επαναλήψεων που πραγματοποιεί κατά την εκτέλεσή της. Με τη λήξη των επαναλήψεων λόγω επίτευξης της επιθυμητής ακρίβειας, τυπώνεται το πλήθος επαναλήψεων που εκτελέστηκαν και η προσέγγιση της ρίζας που υπολογίστηκε.

Αλγόριθμος για τη μέθοδο Τέμνουσας

Η συνάρτηση που υλοποιήθηκε για την εκτέλεση του αλγορίθμου που εφαρμόζει τη μέθοδο της τέμνουσας είναι η `secant_root_estimation()`.

Η συνάρτηση αρχικοποιεί το x_{n-1} χρησιμοποιώντας το πρώτο στοιχείο του πεδίου ορισμού $[-2,2]$ της $f(x)$, δηλαδή το -2 . Επιπρόσθετα, αρχικοποιεί το x_n χρησιμοποιώντας το τελευταίο στοιχείο του πεδίου ορισμού $[-2,2]$ της $f(x)$, δηλαδή το 2 .

Ακολουθώντας, υπολογίζει διαδοχικά το x_{n+1} μέχρις ότου η απόλυτη τιμή της διαφοράς $x_{n+1} - x_n$ να είναι μικρότερη ή ίση του καθορισμένου αποδεκτού σφάλματος (ακρίβεια), δηλαδή $|x_{n+1} - x_n| \leq t_err$. Στο τέλος κάθε επανάληψης, η τιμή του q_{n-1} ανανεώνεται με την τιμή του q_n , ενώ η τιμή του q_n ανανεώνεται με την τιμή του q_{n+1} . Η συνάρτηση καταγράφει σε μία μεταβλητή `iter_count` το πλήθος επαναλήψεων που πραγματοποιεί κατά την εκτέλεσή της. Με τη λήξη των επαναλήψεων λόγω επίτευξης της επιθυμητής ακρίβειας, τυπώνεται το πλήθος επαναλήψεων που εκτελέστηκαν και η προσέγγιση της ρίζας που υπολογίστηκε.

Εκτέλεση των αλγορίθμων και αποτελέσματα

Από την εκτέλεση του αλγορίθμου για τη μέθοδο διχοτόμησης, προκύπτει ως λύση η ακριβής ρίζα 0.0 της $f(x)$.

Από την εκτέλεση του αλγορίθμου για τη μέθοδο Newton-Raphson, προκύπτει ως λύση η προσεγγιστική ρίζα -1.19762372213359 με την εκτέλεση 7 επαναλήψεων.

Από την εκτέλεση του αλγορίθμου για τη μέθοδο της τέμνουσας, προκύπτει ως λύση η προσεγγιστική ρίζα 1.5301335105281786 με την εκτέλεση 10 επαναλήψεων.

Άσκηση 2

Γενικές παρατηρήσεις

Ο κώδικας που έχει συνταχθεί για την άσκηση 2 βρίσκεται εντός του φακέλου “Code/task-2” .

Οι ζητούμενοι αλγόριθμοι που εφαρμόζουν τις τεχνικές προσέγγισης ρίζας με την τροποποιημένη μέθοδο Newton-Raphson, την τροποποιημένη μέθοδο διχοτόμησης και την τροποποιημένη μέθοδο της τέμνουσας, έχουν υλοποιηθεί εντός του αρχείου “modified_root_estimation_algorithms.py” Επιπρόσθετα, εντός του αρχείου f_function.py έχουν δημιουργηθεί οι συναρτήσεις calculate_f(x), calculate_der_1_f(x) και calculate_der_2_f(x). Οι συναρτήσεις αυτές δέχονται ως όρισμα έναν αριθμό ξ και επιστρέφουν την ρίζα της $f(x)$, $f'(x)$ και $f''(x)$ αντίστοιχα, για την δοθείσα συνάρτηση $f(x)$ της εκφώνησης της άσκησης 2. Τέλος, με δεδομένο πως η ζητούμενη ακρίβεια ορίζεται στο 5ο δεκαδικό ψηφίο, η μεταβλητή t_err στο αρχείο “root_estimation_algorithms.py” αρχικοποιείται με την τιμή t_err = 0.000005 (ακρίβεια 5ου δεκαδικού ψηφίου με στρογγυλοποίηση).

Αλγόριθμος για την τροποποιημένη μέθοδο Newton-Raphson

Η συνάρτηση που υλοποιήθηκε για την εκτέλεση του αλγορίθμου που εφαρμόζει τη μέθοδο Newton-Raphson είναι η `modified_newton_raphson_root_estimation(limit_of_iterations=No`

Η συνάρτηση αρχικοποιεί το x_n χρησιμοποιώντας το πρώτο στοιχείο του πεδίου ορισμού $[-2,2]$ της $f(x)$, δηλαδή το -2. Ακολουθώντας, υπολογίζει διαδοχικά το x_{n+1} μέχρις ότου η απόλυτη τιμή της διαφοράς $x_{n+1}-x_n$ να είναι μικρότερη ή ίση του καθορισμένου αποδεκτού σφάλματος (ακρίβεια), δηλαδή $|x_{n+1} - x_n| \leq t_err$. Η συνάρτηση καταγράφει σε μια μεταβλητή iter_count το πλήθος επαναλήψεων που εκτελεί κατά την εκτέλεσή της. Με τη λήξη των επαναλήψεων λόγω επίτευξης της επιθυμητής ακρίβειας, τυπώνεται το πλήθος επαναλήψεων που εκτελέστηκαν και η προσέγγιση της ρίζας που υπολογίστηκε.

Αλγόριθμος για την τροποποιημένη μέθοδο διχοτόμησης

Η συνάρτηση που υλοποιήθηκε για την εκτέλεση του αλγορίθμου που εφαρμόζει την τροποποιημένη μέθοδο της διχοτόμησης είναι η `modified_partitioning_root_estimation()`. Αρχικά, η συνάρτηση υπολογίζει το πλήθος των απαιτούμενων επαναλήψεων της μεθόδου διχοτόμησης για την επίτευξη της ζητούμενης ακρίβειας. Στη συνέχεια, ξεκινά να εκτελεί το ορισμένο πλήθος επαναλήψεων, σταματώντας σε περίπτωση που βρει ακριβή ρίζα της $f(x)$ και τυπώνοντας τη ρίζα, ή συνεχίζοντας μέχρι να ολοκληρώσει το ορισμένο πλήθος επαναλήψεων, τυπώνοντας τελικά την προσέγγιση της ρίζας και το πλήθος επαναλήψεων που απαιτήθηκαν για την εκτίμησή της.

Κατά την εκτέλεση των επαναλήψεων, η επιλογή του τυχαίου σημείου m που χρησιμοποιείται στον υπολογισμό της προσεγγιστικής τιμής $f(m)$ γίνεται με

χρήση της γεννήτριας τυχαίων αριθμών της βιβλιοθήκης ρανδομ που είναι προεγκατεστημένη στην Πψτην. Συγκεκριμένα, η εντολή `random.uniform(Ffunction.calc_space[0], Ffunction.calc_space[1])` επιλέγει σε κάθε της εκτέλεση έναν τυχαίο δεκαδικό αριθμό στο διάστημα $[-2,2]$.

Αλγόριθμος για την τροποποιημένη μέθοδο τέμνουσας

Η συνάρτηση που υλοποιήθηκε για την εκτέλεση του αλγορίθμου που εφαρμόζει την τροποποιημένη μέθοδο της τέμνουσας είναι η `“modified_secant_root_estimation()”`.

Η συνάρτηση αρχικοποιεί το x_n χρησιμοποιώντας το πρώτο στοιχείο του πεδίου ορισμού $[-2,2]$ της $f(x)$, δηλαδή το -2 . Επιπρόσθετα, αρχικοποιεί το x_{n+1} χρησιμοποιώντας το σημείο $x = 0.1$, και το x_{n+2} χρησιμοποιώντας το τελευταίο στοιχείο του πεδίου ορισμού $[-2,2]$ της $f(x)$, δηλαδή το 2 .

Ακολουθώντας, υπολογίζει διαδοχικά το x_{n+3} μέχρις ότου η απόλυτη τιμή της διαφοράς $x_{n+3} - x_{n+2}$ να είναι μικρότερη ή ίση του καθορισμένου αποδεκτού σφάλματος (ακρίβεια), δηλαδή $|x_{n+3} - x_{n+2}| \leq t_err$. Η συνάρτηση καταγράφει σε μια μεταβλητή `υπαδε_ξ_ινδ` ποιό από τα x_n, x_{n+1}, x_{n+2} πρέπει να ανανεωθεί κάθε φορά παίρνοντας την τιμή του x_{n+3} , ανάλογα με το ποιό από τα 3 σημεία προαναφερθέντα σημεία είναι παλιότερο. Επίσης, μια μεταβλητή `iter_count` αξιοποιείται για την καταγραφή του πλήθους επαναλήψεων που πραγματοποιούνται κατά την εκτέλεση της μεθόδου.

Με τη λήξη των επαναλήψεων λόγω επίτευξης της επιθυμητής ακρίβειας, τυπώνεται το πλήθος επαναλήψεων που εκτελέστηκαν και η προσέγγιση της ρίζας που υπολογίστηκε.

Εκτέλεση των αλγορίθμων και αποτελέσματα

Ερώτημα 1.

Από την εκτέλεση του αλγορίθμου για την τροποποιημένη μέθοδο Newton-Raphson, προκύπτει ως λύση η προσεγγιστική ρίζα -1.3812984883894897 με την εκτέλεση 4 επαναλήψεων.

Από την εκτέλεση του αλγορίθμου για την τροποποιημένη μέθοδο διχοτόμησης, προκύπτει ως λύση η προσεγγιστική ρίζα -1.3811460463149443 με την εκτέλεση 19 επαναλήψεων. Σημειώνεται σε αυτό το σημείο πως λόγω της χρήσης γεννήτριας τυχαίων αριθμών, το αποτέλεσμα αυτό συχνά υφίσταται τυχαίες διακυμάνσεις σε κάθε εκτέλεση του αλγορίθμου, όσον αφορά την τιμή της προσεγγιστικής ρίζας.

Από την εκτέλεση του αλγορίθμου για την τροποποιημένη μέθοδο της τέμνουσας, προκύπτει ως λύση η προσεγγιστική ρίζα 0.20518292468887597 με την εκτέλεση 6 επαναλήψεων.

Ερώτημα 2.

Εκτελώντας 10 επαναλήψεις του τροποποιημένου αλγορίθμου διχοτόμησης, λαμβάνουμε τα παρακάτω αποτελέσματα:

1η επανάληψη: Finished after 19 approximation iterations. Approximation is: -1.381790712760268

2η επανάληψη: Finished after 19 approximation iterations. Approximation is: -1.381475154120745

3η επανάληψη: Finished after 19 approximation iterations. Approximation is: 1.9999500173012472

4η επανάληψη: Finished after 19 approximation iterations. Approximation is: 1.999999979070466

5η επανάληψη: Finished after 19 approximation iterations. Approximation is: -1.3830710263393167

6η επανάληψη: Finished after 19 approximation iterations. Approximation is: -1.3813186273722509

7η επανάληψη: Finished after 19 approximation iterations. Approximation is: -1.3799733240691971

8η επανάληψη: Finished after 19 approximation iterations. Approximation is: -1.3814260841319628

9η επανάληψη: Finished after 19 approximation iterations. Approximation is: -1.3817409842964017

10η επανάληψη: Finished after 19 approximation iterations. Approximation is: 1.9999999593232018

Παρατηρούμε ότι και στις 10 εκτελέσεις του αλγορίθμου, η σύγκλιση στην προσεγγιστική ρίζα προκύπτει μέσα από 19 επαναλήψεις. Συνεπώς, ο αριθμός επαναλήψεων παραμένει σταθερός.

Ερώτημα 3.

Εκτελώντας πειραματικά τους αλγορίθμους προσέγγισης ρίζας με τις κλασσικές μεθόδους, έχουμε τα εξής αποτελέσματα:

Ο αλγόριθμος της μεθόδου διχοτόμησης συγκλίνει μετά από 1 μόλις επανάληψη,

αφού βρίσκει την ακριβή ρίζα 0.0 της $f(x)$

Ο αλγόριθμος της μεθόδου Newton-Raphson συγκλίνει σε προσεγγιστική ρίζα μετά από 7 επαναλήψεις.

Ο αλγόριθμος της μεθόδου της τέμνουσας συγκλίνει σε προσεγγιστική ρίζα μετά από 10 επαναλήψεις.

Εκτελώντας πειραματικά τους αλγορίθμους προσέγγισης ρίζας με τις τροποποιημένες μεθόδους αντίστοιχα, έχουμε τα εξής αποτελέσματα:

Ο τροποποιημένος αλγόριθμος της μεθόδου διχοτόμησης συγκλίνει σε προσεγγιστική ρίζα μετά από 19 επαναλήψεις.

Ο τροποποιημένος αλγόριθμος της μεθόδου Newton-Raphson συγκλίνει σε προσεγγιστική ρίζα μετά από 4 επαναλήψεις.

Ο τροποποιημένος αλγόριθμος της μεθόδου της τέμνουσας συγκλίνει σε προσεγγιστική ρίζα μετά από 6 επαναλήψεις.

Συμπεραίνουμε λοιπόν πως ως προς την ταχύτητα σύγκλισης:

Η τροποποιημένη μέθοδος διχοτόμησης είναι 19 φορές πιο αργή από την κλασσική.

Η τροποποιημένη μέθοδος Newton-Raphson είναι περίπου 42.8% γρηγορότερη από την κλασσική

Η τροποποιημένη μέθοδος της τέμνουσας είναι 20% γρηγορότερη από την κλασσική.

Άσκηση 3

Γενικές παρατηρήσεις

Ο κώδικας που έχει συνταχθεί για την άσκηση 3 βρίσκεται εντός του φακέλου “Code/task-3” .

Ερώτημα 1

Το αρχείο κώδικα είναι το `pa.lu_gauss.py`. Για την υλοποίηση της λύσης του ερωτήματος, αξιοποιήθηκε η μέθοδος αποσύνθεσης πινάκων “Doolittle’s LUP” .

Η κύρια συνάρτηση της λύσης είναι η `gauss(a_matrix, b_vector)`, η οποία δέχεται ως ορίσματα έναν τετραγωνικό πίνακα `a_matrix` διαστάσεων $n \times n$ και ένα διάνυσμα `b_vector` μεγέθους n με σκοπό την επίλυση του συστήματος $Ax=b$. Τελικά, η συνάρτηση `gauss` επιστρέφει το διάνυσμα αγνώστων x μεγέθους n που περιέχει τις τιμές της λύσης του συστήματος.

Για την εκτέλεση των υπολογισμών με την μέθοδο επίλυσης γραμμικών συστημάτων $PA=LU$ υλοποιήθηκε μια σειρά βοηθητικών συναρτήσεων, η οποία θα περιγραφεί συνοπτικά στη συνέχεια. Η συνάρτηση `multiply_matrices(m1_matrix, m2_matrix)` δέχεται ως όρισμα τον πίνακα `m1_matrix` διαστάσεων $n \times k$ και τον πίνακα `m2_matrix` διαστάσεων $k \times m$ ώστε εκτελώντας τους κατάλληλους υπολογισμούς, να επιστρέψει έναν πίνακα που αποτελεί αποτέλεσμα του πολλαπλασιασμού των πινάκων `m1_matrix` και `m2_matrix`.

Η συνάρτηση `perform_pivoting(m_matrix)` δέχεται ως όρισμα έναν πίνακα `m_matrix`, στον οποίο πραγματοποιεί τις κατάλληλες 'ανταλλαγές' μεταξύ των γραμμών του πίνακα, ώστε με το πέρας της διαδικασίας, το μεγαλύτερο στοιχείο κατά απόλυτη τιμή κάθε στήλης του πίνακα να βρίσκεται πάνω στην κύρια διαγώνιο του πίνακα `m_matrix`.

Αξίζει να σημειωθεί ότι οι αλλαγές στις σειρές του `m_matrix` γίνονται *in place*, δηλαδή μεταβάλλεται άμεσα το περιεχόμενο του πίνακα που δίνεται κατά την κλήση της συνάρτησης ως όρισμα, αφού στην Python οι λίστες είναι μεταλλάξιμος (*mutable*) τύπος δεδομένων.

Τέλος, η συνάρτηση επιστρέφει τον πίνακα αλλαγών γραμμών (*pivoting matrix* - *P matrix*) που προκύπτει από την διαδικασία αλλαγών γραμμών του `m_matrix`.

Η συνάρτηση `perform_pa_lu_decomposition(a_matrix)` λαμβάνει ως όρισμα έναν τετραγωνικό πίνακα `a_matrix` διαστάσεων $n \times n$. Προκειμένου να εκτελεστεί η αποσύνθεση $PA=LU$ στον πίνακα, ακολουθείται μια σειρά βημάτων. Αρχικά, δημιουργούνται οι πίνακες `L_matrix` και `u_matrix` διαστάσεων $n \times n$. Στη συνέχεια, αντιγράφεται (με χρήση της εντολής `copy()` αφού οι λίστες είναι *mutable* τύπος δεδομένων) ο πίνακας `a_matrix` σε μια νέα μεταβλητή `pa_matrix`. Η μεταβλητή `pa_matrix` δίνεται ως όρισμα σε κλήση της συνάρτησης `perform_pivoting(pa_matrix)`, η οποία κάνει τις κατάλληλες ανταλλαγές μεταξύ των σειρών της `pa_matrix` *in-place*, δηλαδή οι αλλαγές γίνονται απευθείας στη μεταβλητή που δίνεται ως όρισμα. Τελικά η `perform_pivoting(pa_matrix)` έχοντας ολοκληρώσει την επεξεργασία της `pa_matrix`, επιστρέφει και τον πίνακα αλλαγών γραμμών (*pivot matrix*), που εκχωρείται στην μεταβλητή `p_matrix`.

Ακολούθως, σχηματίζεται ο άνω τριγωνικός πίνακας U σύμφωνα με τον παρακάτω τύπο:

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} u_{kj} l_{ik}$$

ενώ ο κάτω τριγωνικός πίνακας L σχηματίζεται σύμφωνα με τον τύπο:

$$l_{ij} = \frac{1}{u_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} u_{kj} l_{ik} \right)$$

Τέλος, η συνάρτηση επιστρέφει ως αποτέλεσμα τους πίνακες `p_matrix`, `l_matrix` και `u_matrix` που αντιπροσωπεύουν τους πίνακες P, L και U αντίστοιχα, οι οποίοι προκύπτουν μέσω της αποσύνθεσης $PA = LU$.

Ερώτημα 2

Το αρχείο κώδικα είναι το `cholesky.py`. Για την υλοποίηση της λύσης του ερωτήματος, έχει δημιουργηθεί η συνάρτηση (`perform_cholesky_decomposition(a_matrix)`). Η συνάρτηση δέχεται ως όρισμα έναν συμμετρικό και θετικά ορισμένο πίνακα `a_matrix` διαστάσεων `nxn` στον οποίο θα πραγματοποιηθεί ακολούθως η αποσύνθεση Cholesky. Αρχικά δημιουργείται ο πίνακας `l_matrix` διαστάσεων `nxn` και αρχικοποιείται με 0.0 σε κάθε θέση του. Στη συνέχεια, γίνεται ο υπολογισμός της αποσύνθεσης Cholesky σύμφωνα με τους παρακάτω τύπους:

$$l_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2}$$

και

$$l_{ik} = \frac{1}{l_{kk}} \left(a_{ik} - \sum_{j=1}^{k-1} l_{ij} l_{kj} \right)$$

Τελικά, η συνάρτηση επιστρέφει ως αποτέλεσμα έναν κάτω τριγωνικό πίνακα (πίνακας L), ο οποίος αποτελεί το αποτέλεσμα της αποσύνθεσης Cholesky.

Ερώτημα 3

Το αρχείο κώδικα είναι το `gauss_seidel.py`. Για την υλοποίηση της λύσης του ερωτήματος, έχει δημιουργηθεί η βοηθητική συνάρτηση `calculate_vector_diff(vector_1, vector_2)` καθώς και η κύρια συνάρτηση `generate_a_matrix(n)`. Η συνάρτηση `calculate_vector_diff(vector_1, vector_2)` δέχεται ως ορίσματα τα διανύσματα `vector_1` και `vector_2`, υπολογίζει τη διαφορά τους και την επιστρέφει ως ένα νέο διάνυσμα αποτελέσματος. Η συνάρτηση `generate_a_matrix(n)` δέχεται ως όρισμα έναν αριθμό `n` που αποτελεί τη διάσταση του πίνακα A (`nxn`). Αρχικά δημιουργείται ο πίνακας `a_matrix` διάστασης `nxn` και αρχικοποιείται με 0 σε κάθε θέση του. Στη συνέχεια, σύμφωνα με τους κανόνες που περιγράφονται στην εκφώνηση για τον τρόπο συμπλήρωσης των θέσεων του πίνακα, γίνεται

τροποποίηση των κατάλληλων θέσεων του `a_matrix` με τις τιμές 5 και -2. Επιπρόσθετα, το διάνυσμα `b_vector` μεγέθους `n` αρχικοποιείται σύμφωνα με τους κανόνες της εκφώνησης (3 στην πρώτη και τελευταία θέση του, 1 σε όλες τις υπόλοιπες), ενώ το διάνυσμα `x_vector` μεγέθους `n` αρχικοποιείται με 0 σε όλες τις θέσεις του (θεωρώ αρχική τιμή όλων των στοιχείων του διανύσματος το 0).

Για τον υπολογισμό της προσεγγιστικής λύσης του προβλήματος, υπολογίζονται επαναληπτικά οι τιμές όλων των στοιχείων του `x_vector`, ενώ στην αρχή κάθε επανάληψης αποθηκεύεται η κατάσταση του `x_vector` στο τέλος της προηγούμενης επανάληψης (εντολή `old_x_vector = x_vector.ζοψ()`, προκειμένου να ικανοποιηθούν οι ανάγκες του μαθηματικού τύπου του θεωρήματος Gauss-Seidel. Συγκεκριμένα, ο τύπος Gauss-Seidel που υλοποιήθηκε σε κώδικα, είναι ο εξής:

$$x_i^{(m+1)} = \frac{1}{a_{ij}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(m+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(m)} \right)$$

1 Άσκηση 1 (Λύση)

ABΓΔΕΖΗΘΙΚ $\kappa\lambda\mu\nu\xi\omicron\pi\rho\varsigma$

2 Άσκηση 2 (Λύση)

Normal *Italics* **Bold**
Emphasized Underlined

3 Άσκηση 3 (Λύση)

$$a^2 + b^2 = c^2$$

$$e^{i\pi} = -1$$

$$\pi = \frac{c}{d}$$

$$\frac{d}{dx} \int_a^x f(s) ds = f(x)$$

$$f(x) = \sum_{i=0}^{\infty} \frac{f^{(i)}(0)}{i!} x^i$$

$$\mathbf{Ax} = \mathbf{b}$$

$$||x + y|| \leq ||x|| + ||y||$$

$$\mathbf{I} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1)$$

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$\mathbf{I} = \begin{Bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{Bmatrix}, \quad \mathbf{I} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}, \quad \mathbf{I} = \left\| \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \right\| \quad (3)$$

4 Άσκηση 4 (Λύση)

<i>Τέφας</i>	<i>2</i>	<i>3</i>
<i>Πήτας</i>	<i>5</i>	<i>6</i>
<i>Λάσκαρης</i>	<i>8</i>	<i>9</i>

<i>Κοτρόπουλος</i>	<i>2</i>	<i>3</i>
<i>Πήτας</i>	<i>5</i>	<i>6</i>
<i>Νικολαΐδης</i>	<i>8</i>	<i>9</i>

<i>1</i>	<i>2</i>	<i>3</i>
<i>4</i>	<i>5</i>	<i>6</i>
<i>7</i>	<i>8</i>	<i>9</i>

<i>1</i>	<i>2</i>	<i>3</i>
<i>4</i>	<i>5</i>	<i>6</i>
<i>7</i>	<i>8</i>	<i>9</i>

Μέλη ΔΕΠ Πληροφορικής		
Λέκτορες	VD	Δραζιώτης Κωνσταντίνος
Επικουροι	LN	Λάσκαρης Νικόλαος
	TG	Τσουμάκας Γρηγόριος
Αναπληρωτές	TA	Τέφας Αναστάσιος
	PN	Πλέρος Νίκος
	PA	Παπαδόπουλος Απόστολος
Καθηγητές	KC	Κοτρόπουλος Κωνσταντίνος
	PI	Πήτας Ιωάννης
	VI	Βλαχάβας Ιωάννης

5 Άσκηση 5 (Λύση)

- Τέφας
- Μπουζάς
- Μπρούζα
- Λάσκαρης
- Κοτρόπουλος
- Πήτας
- Νικολαΐδης

1. Τέφας
2. Μπουζάς
3. Μπρούζα
4. Λάσκαρης
5. Κοτρόπουλος
6. Πήτας
7. Νικολαΐδης

- (α) Τέφας
- (β) Μπουζάς
- (γ) Μπρούζα
- (δ) Λάσκαρης

(ε) Κοτρόπουλος

(ζ) Πήτας

(η) Νικολαΐδης