ΑΡΙΘΜΗΤΙΚΗ ΓΡΑΜΜΙΚΗ ΑΛΓΕΒΡΑ

ΠΑΠΑΓΙΑΝΝΗ ΙΩΑΝΝΑ

ΓΕΝΙΚΑ:

Η πρώτη εργαστηριακή άσκηση αφορά την επίλυση δυο γραμμικών συστημάτων με την μέθοδο Cholesky. Η μέθοδος Cholesky αναλύει τους A1, A2 σε δύο πίνακες L και L' όπου ο L είναι ένας κάτω τριγωνικός και ο L' είναι ο ανάστροφος του L (άνω τριγωνικός).

Στην συνέχεια το σύστημα μετατρέπεται σε δυο συστήματα:
1. L'x=y
2. Ly=b

Αρχικά λύνουμε το 2. σύστημα για τον υπολογισμό του y και στην συνέχεια επιλύουμε το 1. για τον υπολογισμό του x.

ΣΥΝΑΡΤΗΣΕΙΣ:

def decomp_cholesky(A):
Η συνάρτηση αυτή επιστρέφει τον πίνακα L (κάτω τριγωνικό) τον οποίο υπολογίζει με την μέθοδο Cholesky.

def solve_lower(L,b):
Παίρνει σαν είσοδο ένα κάτω τριγωνικό πίνακα και έναν μονοδιάστατο πίνακα και επιστρέφει την λύση του συστήματος 2. δηλαδή το y. Η λύση γίνεται με αντικατάσταση από κάτω προς τα πάνω.

def solve_upper(L,b):
Παίρνει σαν είσοδο ένα κάτω τριγωνικό πίνακα και έναν μονοδιάστατο πίνακα και επιστρέφει την λύση του συστήματος 1. δηλαδή το x. Τον κάτω τριγωνικό τον μετατρέπει σε άνω τριγωνικό για να κάνει την οπισθοδρόμηση και υπολογίσει την λύση. Ο άνω τριγωνικός προκύπτει από την αλλαγή των παραμέτρων i, j του κάτω τριγωνικού L (L**[i][j]** ήταν στον κώδικα του *def solve_lower(L,b)* και U**[j][i]** στον κώδικα του *def solve_upper(U, y)*).
Η λύση γίνεται με αντικατάσταση από πάνω προς τα κάτω.

def solve_cholesky(L,b):
Αρχικά λύνει το 2. σύστημα για τον υπολογισμό του y καλώντας την *def solve_lower(L,b)* με ορίσματα L, b και στην συνέχεια επιλύει το 1. για τον υπολογισμό του x καλώντας την *def solve_upper(U,y)* με ορίσματα L, y . (όπως εξηγήσαμε και παραπάνω παίρνει ως όρσιμα L και μέσα στην συναρτηση τον χρησιμοποιεί ως L').

if __name__ == '__main__':
Η main() συνάρτηση δημιουργεί τους αρχικούς πίνακες A1, A2 , b1, b2 και έπειτα καλεί με την σειρά την *def decomp_cholesky(A)* και solve_cholesky(L,b) για τα δυο συστήματα αντίστοιχα και μετά εκτυπώνει τις 10 μεσαίες τιμές του x.

ΣΥΜΠΕΡΑΣΜΑΤΑ:

Για N=100,1000,10000 αλλάζουμε το N στην εκχώρηση. Παρατηρούμε ότι οι λύσεις είναι διαφορετικές για τους πίνακες αλλά είναι κοντά στην πραγματική που είναι όλες οι τιμές ίσες με 1. Αυτό γίνεται γιατί οι αριθμοί αποθηκεύονται με συγκεκριμένη ακρίβεια στον υπολογιστή. Οι δύο πίνακες επηρεάζονται διαφορετικά γιατί έχουν διαφορετικό δείκτη κατάστασης που καθορίζει το πώς οι διαταραχές του πίνακα A επηρεάζουν τη λύση και ο οποίος προσδιορίζει τη μέγιστη δυνατή μεταβολή του σχετικού σφάλματος των αποτελεσμάτων σε σχέση με το σχετικό σφάλμα δεδομένων. Ο A1 χάνει ακρίβεια.

Ας αφήσουμε προς στιγμήν τον πίνακα A σταθερό και ας μεταβάλλουμε το διάνυσμα στήλη b. Καλή κατάσταση έχουμε όταν αλλάζουμε λίγο το b τότε αλλάζει λίγο το x. Κακή έχουμε όταν αλλάζουμε λίγο το b και το x αλλάζει πολύ. Αλλάζουμε λοιπόν το πρώτο και το τελευταίο στοιχείο του b1 και αντιστοιχα b2 (και τα δύο τα αυξάνουμε κατά 1).
Παρατηρούμε ότι ο A_2 έχει καλή κατάσταση ενώ ο A_1 έχει κακή.

```
A1 = np.zeros((N,N), float)
A2 = np.zeros((N,N), float)

b1 = np.zeros((N,1), float)
b2 = np.ones((N,1), float)

#Filling arrays with the correspondant va

np.fill_diagonal(A1, 6)
np.fill_diagonal(A1[1:], -4)
np.fill_diagonal(A1[:, 1:], -4)
np.fill_diagonal(A1[2:], 1)
np.fill_diagonal(A1[:, 2:], 1)

np.fill_diagonal(A2, 7)
np.fill_diagonal(A2[1:], -4)
np.fill_diagonal(A2[:, 1:], -4)
np.fill_diagonal(A2[2:], 1)
np.fill_diagonal(A2[:, 2:], 1)

b1[0] = 4
b1[1] = -1
b1[-2] = -1
b1[-1] = 4

b2[0] = 5
b2[1] = 0
b2[-2] = 0
b2[-1] = 5
```

```
N:  1000
A1 x = b1
 Ten median x are:
[[250.9800]
 [250.9880]
 [250.9940]
 [250.9980]
 [251.0000]
 [251.0000]
 [250.9980]
 [250.9940]
 [250.9880]
 [250.9800]]
A2 x = b2
 Ten median x are:
[[1.0000]
 [1.0000]
 [1.0000]
 [1.0000]
 [1.0000]
 [1.0000]
 [1.0000]
 [1.0000]
 [1.0000]
 [1.0000]]
```

ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΩΝ ΛΥΣΕΩΝ, ΔΕΚΑ ΜΕΣΣΑΙΩΝ Χ:

Για το σύστημα $A_1 x = b_1$

N = 100	N = 1000	N = 10000
1.0000	1.0004	0.9998
1.0000	1.0004	0.9998
1.0000	1.0004	0.9998
1.0000	1.0004	0.9999
1.0000	1.0004	0.9996
1.0000	1.0004	0.9999
1.0000	1.0004	0.9999
1.0000	1.0004	0.9999
1.0000	1.0005	0.9999
1.0000	1.0005	0.9999

Για το σύστημα $A_2 x = b_2$

N = 100	N = 1000	N = 10000
1.0000	1.0000	1.0000
1.0000	1.0000	1.0000
1.0000	1.0000	1.0000
1.0000	1.0000	1.0000
1.0000	1.0000	1.0000
1.0000	1.0000	1.0000
1.0000	1.0000	1.0000
1.0000	1.0000	1.0000
1.0000	1.0000	1.0000
1.0000	1.0000	1.0000

CODE:

```
import numpy as np
import math
from math import sqrt
float_percision = '{:.4f}'.format
np.set_printoptions(formatter={'float_kind':float_percision})

def decomp_cholesky(A):

    n = len(A)
    L = np.zeros((n, n))
```

```
    for i in range(n):
        for j in range(i+1):
            L[i][j] = A[i][j]
            for k in range(j):
                L[i][j] -= L[i][k]*L[j][k]
            if(i == j):
                L[i][j] = sqrt(L[j][j])
            else:
                L[i][j] /= L[j][j]
    return L

def solve_lower(L, b):
    n = len(L)
    y = np.copy(b)
    y[0] = b[0] / L[0][0]
    for i in range(1, n):
        for j in range(i):
            y[i] -= L[i][j]*y[j]
        y[i] /= L[i][i]
    return y

def solve_upper(U, y):
    n = len(U)
    x = np.copy(y)
    for i in range(n-1, -1, -1):
        for j in range(i+1, n):
            x[i] -= U[j][i]*x[j]
        x[i] /= U[i][i]
    return x
```

```
def solve_cholesky(L,b):

    y = solve_lower(L,b)
    x = solve_upper(L,y)
    return x

if __name__ == '__main__':

        N = 1000
        print('N = ', N)

        #Filling N*N array to initialize it
        A1 = np.zeros((N,N), float)
        A2 = np.zeros((N,N), float)

        b1 = np.zeros(N, float)
        b2 = np.ones(N, float)

        #Fill arrays with the correspondant values
        np.fill_diagonal(A1, 6)
        np.fill_diagonal(A1[1:], -4)
        np.fill_diagonal(A1[:, 1:], -4)
        np.fill_diagonal(A1[2:], 1)
        np.fill_diagonal(A1[:, 2:], 1)

        np.fill_diagonal(A2, 7)
        np.fill_diagonal(A2[1:], -4)
        np.fill_diagonal(A2[:, 1:], -4)
        np.fill_diagonal(A2[2:], 1)
        np.fill_diagonal(A2[:, 2:], 1)

        b1[0] = 3
        b1[1] = -1
        b1[-2] = -1
        b1[-1] = 3

        b2[0] = 4
        b2[1] = 0
        b2[-2] = 0
        b2[-1] = 4

        L = decomp_cholesky(A1)
        x = solve_cholesky(L, b1)

        print('A1 x = b1 \n Ten median x are:')
        ml = len(x) // 2 - 5
        mu = len(x) // 2 + 5
        print(x[ml : mu])

        L = decomp_cholesky(A2)
        x = solve_cholesky(L, b2)

        print('A2 x = b2 \n Ten median x are:')
        ml = len(x) // 2 - 5
        mu = len(x) // 2 + 5
        print(x[ml : mu])
```