

Εξαιρετική παρουσίαση και προσπάθεια.  
Μαζί με τους κώδικες θα ήταν καλό να παραθέτεις και πειράματα ώστε να ελέγχονται και να αναδεικνύονται οι ισχυρισμοί σου.



ΠΑΝΕΠΙΣΤΗΜΙΟ  
ΠΑΤΡΩΝ  
UNIVERSITY OF PATRAS

### Εργασία 3

Χαράλαμπος Αναστασίου  
ΑΜ: 1093316

Τμήμα Μηχανικών Υπολογιστών και Πληροφορικής  
Πανεπιστήμιο Πατρών

Νοέμβριος 2024

### Πίνακας Περιεχομένων

1	Εισαγωγή	2
2	Απαντήσεις Ερωτημάτων	2
2.1	Ερώτημα 1	2
2.2	Ερώτημα 2	3
2.3	Ερώτημα 3	4
2.4	Ερώτημα 4	4
2.5	Ερώτημα 5	6
2.6	Ερώτημα 6	7
2.7	Ερώτημα 7	7
2.8	Ερώτημα 8	8

Ερώτημα 1:

Όσον αφορά το δεύτερο μέρος, αξίζει να δούμε τη μορφή του  $L^{-1}$  για την περίπτωση  $n = 6$ :

```
n=8; L=eye(n)-tril(ones(n),-1);
M = inv(L)

M = 8x8
    1     0     0     0     0     0     0     0
    1     1     0     0     0     0     0     0
    2     1     1     0     0     0     0     0
    4     2     1     1     0     0     0     0
    8     4     2     1     1     0     0     0
   16     8     4     2     1     1     0     0
   32    16     8     4     2     1     1     0
   64    32    16     8     4     2     1     1
```

Παρατηρούμε ότι το μητρώο είναι κάτω τριγωνικό Toeplitz με τιμές που καθορίζονται από το διάνυσμα  $[1, a, a^2, \dots, a^{n-2}]$  όπου  $a = 2$  (αποδείξτε το). Για να υπολογίσουμε τη νόρμα Frobenius, παρατηρούμε το  $M \odot M$ :

```
M.*M

ans = 8x8
    1     0     0     0     0     0     0     0
    1     1     0     0     0     0     0     0
    4     1     1     0     0     0     0     0
   16     4     1     1     0     0     0     0
   64    16     4     1     1     0     0     0
  256    64    16     4     1     1     0     0
 1024   256    64    16     4     1     1     0
 4096  1024   256    64    16     4     1     0
```

Αθροίζοντας κατά μήκος των διαγωνίων και θέτοντας  $r = a^2$ , το τετράγωνο της νόρμας Frobenius θα είναι:

$$f = n + (n - 1) + (n - 2)r + (n - 3)r^2 + \dots + (n - (n - 1))r^{n-2}$$

Το συγκεκριμένο άθροισμα μπορεί να υπολογιστεί ως εξής:

$$f = n + n(1 + r + r^2 + \dots + r^{n-2}) - \sum_{j=1}^{n-1} jr^{j-1} = n + n \frac{r^{n-1} - 1}{r - 1} - \frac{d}{dr} \frac{r^{n-1} - 1}{r - 1}$$

Παρατηρούμε ότι το τελευταίο άθροισμα είναι η παράγωγος του μεσαίου όρου (στοιχείο προς στοιχείο), επομένως προκύπτει το εξής:

$$f = n - \frac{r^{n-1} n (r - 1) - (r^n - 1)}{(r - 1)^2} + \frac{n (r^{n-1} - 1)}{r - 1}$$

Επαληθεύουμε για όποια περίπτωση επιθυμούμε:

```
f = @(n,r)n+n*(r^(n-1)-1)/(r-1)-((r-1)*n*r^(n-1)-(r^n-1))/(r-1)^2;
f(n,4)
```

```
ans =
7287
```

```
norm(M, 'fro')^2
```

```
ans =
7.2870e+03
```

# 1 Εισαγωγή

Ερωτήματα που απαντήθηκαν: 1, 2, 3, 4, 5, 6

## 2 Απαντήσεις Ερωτημάτων

### 2.1 Ερώτημα 1

(GvL A3.1.2) Υποθέτουμε ότι το  $L = I_n - N$  είναι μοναδιαίο κάτω τριγωνικό, με  $N \in \mathbb{R}^{n \times n}$ . Να δείξετε ότι:

$$L^{-1} = I_n + N + N^2 + \dots + N^{n-1}.$$

Ποια είναι η τιμή του  $\|L^{-1}\|_F$  αν έχουμε  $N_{ij} = 1$  για κάθε  $i > j$ ?

#### ΑΠΑΝΤΗΣΗ:

Για να αποδείξουμε ότι  $L^{-1} = I_n + N + N^2 + \dots + N^{n-1}$ , αρκεί να εξασφαλίσουμε ότι ισχύει:

$$LL^{-1} = I_n$$

Έχουμε:

$$LL^{-1} = (I_n - N)(I_n + N + N^2 + \dots + N^{n-1}).$$



Κάνοντας τις πράξεις προκύπτει:

$$LL^{-1} = I_n + N + N^2 + \dots + N^{n-1} - N - N^2 - N^3 - \dots - N^n.$$

Μπορούμε να παρατηρήσουμε ότι οι όροι  $N, N^2, \dots, N^{n-1}$  και  $-N, -N^2, \dots, -N^{n-1}$  θα ακυρωθούν. Έτσι, απομένει:

$$I_n - N^n.$$



Αλλά  $N^n = 0$  επειδή το  $N$  είναι αυστηρά κάτω τριγωνικό. Το γεγονός ότι είναι αυστηρά κάτω τριγωνικό προκύπτει από τον τύπο  $L = I_n - N$  όπου το  $L$  είναι μοναδιαίο κάτω τριγωνικό μητρώο (επομένως προκειμένου να εξασφαλιστεί η μοναδιαία διαγώνιος του  $L$  πρέπει το  $N$  να έχει μηδενική διαγώνιο, άρα να είναι αυστηρά κάτω τριγωνικό). Έτσι, έχουμε:

$$LL^{-1} = I_n.$$

Απεδείχθη επομένως ότι

$$L^{-1} = I_n + N + N^2 + \dots + N^{n-1}.$$

Υπολογισμός του  $\|L^{-1}\|_F$  όταν  $N_{ij} = 1$  για κάθε  $i > j$  :

Αφού  $N$  είναι αυστηρά κάτω τριγωνικό με  $N_{ij} = 1$  για κάθε  $i > j$ , τα στοιχεία του  $N$  είναι ίσα με 1 κάτω από την κύρια διαγώνιο, ενώ όλα τα υπόλοιπα στοιχεία είναι 0.

Ο τύπος που αποδείξαμε δίνει:

$$L^{-1} = I_n + N + N^2 + \dots + N^{n-1}$$

Όμως, καθώς το  $N$  είναι αυστηρά κάτω τριγωνικό, το  $N^k$  για  $k \geq n$  είναι το μηδενικό μητρώο. Έτσι, η σειρά σταματά στο  $N^{n-1}$ .

Η νόρμα Frobenius  $\|A\|_F$  ενός μητρώου  $A$  δίνεται από τον τύπο:

$$\|A\|_F = \sqrt{\sum_{i,j} |a_{ij}|^2}$$

Για να βρούμε την  $\|L^{-1}\|_F$ , πρέπει να υπολογίσουμε το άθροισμα των τετραγώνων των στοιχείων του  $L^{-1}$ .

Λόγω της κατασκευής του  $L^{-1}$ , κάθε στοιχείο κάτω από την κύρια διαγώνιο έχει κάποια συγκεκριμένη τιμή ανάλογα με τη θέση του και τον αριθμό των μη μηδενικών στοιχείων του κάθε  $N^k$  μέχρι  $k = n - 1$ .

Έτσι, για να υπολογιστεί ακριβώς η τιμή του  $\|L^{-1}\|_F$  σε αυτή την ειδική περίπτωση, θα πρέπει να υπολογίσουμε το άθροισμα όλων των στοιχείων σε κάθε δύναμη του  $N$  και στη συνέχεια να λάβουμε το άθροισμα των τετραγώνων αυτών των στοιχείων. Στο επισυνυπτόμενο .mlx αρχείο υπολογίστηκε το  $L^{-1}$  (όπου φαίνεται η χαρακτηριστική δομή του με βάση τα όσα περιέγραψα παραπάνω) και η νόρμα Frobenius.

Μπορείς (καλύτερα) αναλυτικά - βλ. συνημμένο.

## 2.2 Ερώτημα 2

Δίνεται το συμμετρικό μητρώο  $4 \times 4$ :

$$\begin{pmatrix} a_{11} & a_{12} & | & a_{13} & a_{14} \\ a_{12} & a_{22} & | & a_{23} & a_{24} \\ a_{13} & a_{23} & | & a_{33} & a_{34} \\ a_{14} & a_{24} & | & a_{34} & a_{44} \end{pmatrix}$$

Να δείξετε την αποθήκευση του μητρώου στη μορφή RFP (rectangular full packed format).

### ΑΠΑΝΤΗΣΗ:

Στη μορφή RFP, αποθηκεύουμε μόνο τις μοναδικές τιμές του συμμετρικού μητρώου, καθώς οι τιμές πάνω από την κύρια διαγώνιο είναι ίσες με τις αντίστοιχες τιμές κάτω από αυτήν. Θα αποθηκεύσουμε επομένως τα στοιχεία ως εξής:

**Κύρια διαγώνιος:**  $a_{11}, a_{22}, a_{33}, a_{44}$

**Στοιχεία πάνω από την κύρια διαγώνιο:**

$a_{12}, a_{13}, a_{14}, a_{23}, a_{24}, a_{34}$

Προσοχή:

Η αποθήκευση του μητρώου  $A$  στη μορφή RFP θα είναι:

Μορφή RFPF:

$$\text{RFP} = \{a_{11}, a_{22}, a_{33}, a_{44}, a_{12}, a_{13}, a_{14}, a_{23}, a_{24}, a_{34}\}$$

$$\begin{pmatrix} a_{33} & a_{34} \\ a_{11} & a_{44} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \\ a_{14} & a_{24} \end{pmatrix}$$

## 2.3 Ερώτημα 3

Να γράψετε συνάρτηση MATLAB `ro2rpf` που λαμβάνει ως είσοδο ένα συμμετρικό μητρώο που δίνεται σε κανονική αποθήκευση και επιστρέφει το μητρώο σε packed μορφή ως διάνυσμα. Ο κώδικας θα πρέπει να επιτρέπει στο χρήστη να επιλέξει κατά πόσον το packing θα είναι column ή row-major και κατά πόσον αποθηκεύεται το άνω ή το κάτω τριγωνικό τμήμα.

### ΑΠΑΝΤΗΣΗ:

Δες την απάντηση στον κώδικα του .mlx αρχείου

## 2.4 Ερώτημα 4

(GvL A4.2.10) Έστω ότι  $A = I + uu^T$  όπου  $A \in \mathbb{R}^{n \times n}$  και  $\|u\|_2 = 1$ . Να γράψετε ακριβείς τύπους για τη διαγώνιο και την υποδιαγώνιο του παράγοντα Cholesky του  $A$ .

### ΑΠΑΝΤΗΣΗ:



Αρχικά θα εξετάσω γιατί το μητρώο είναι θετικά ορισμένο και άρα μπορώ να εφαρμόσω Cholesky (ΔΕΝ ΖΗΤΕΙΤΑΙ - ΜΟΝΟ ΓΙΑ ΠΡΟΣΩΠΙΚΗ ΚΑΤΑΝΟΗΣΗ).

Σύμφωνα με το Θεώρημα 4.2.3 του GvL, ένα μητρώο  $A$  είναι θετικά ορισμένο αν και μόνο αν το συμμετρικό  $T = \frac{A+A^T}{2}$  έχει θετικές ιδιοτιμές.

Κάνοντας τις πράξεις διαπιστώνω ότι ισχύει:

$$T = \frac{A + A^T}{2} = A$$

Το μητρώο  $I$  είναι το μοναδιαίο, το οποίο είναι θετικά ορισμένο, καθώς έχει μόνο ιδιοτιμές ίσες με 1.

Το μητρώο  $uu^T$  είναι θετικά ημι-ορισμένο (positive semi-definite). Αυτό σημαίνει ότι για κάθε μη μηδενικό διάνυσμα  $x$ , ισχύει ότι:

$$x^T(uu^T)x \geq 0$$

Πράγματι, η τιμή του  $x^T(uu^T)x$  είναι:

$$x^T(uu^T)x = (u^T x)(u^T x) = (u^T x)^2 \geq 0$$

Για να εξετάσουμε αν το  $I + uu^T$  είναι θετικά ορισμένο, θα πρέπει να δούμε τις ιδιοτιμές του:

Οι ιδιοτιμές του  $I + uu^T$  προκύπτουν από τις ιδιοτιμές του  $uu^T$ , αλλά προστίθεται 1 (δηλαδή η ιδιοτιμή του μοναδιαίου μητρώου  $I$ ). Το  $uu^T$  είναι τάξης-1 και έχει μία ιδιοτιμή ίση με 1 (η μοναδική ιδιοτιμή ισούται με τη νόρμα-2 του  $u$ ) και  $n-1$  ιδιοτιμές ίσες με 0. Άρα οι ιδιοτιμές του  $I + uu^T$  θα είναι:

- Μία ιδιοτιμή  $1 + 1 = 2$  (που αντιστοιχεί στην ιδιοτιμή του  $uu^T$  που είναι 1).
- $n - 1$  ιδιοτιμές  $1 + 0 = 1$  (που αντιστοιχούν στις μηδενικές ιδιοτιμές του  $uu^T$ ).

Άρα οι ιδιοτιμές του  $I + uu^T$  είναι 2 και 1 (με πολλαπλότητα  $n-1$ ). ✓

Επειδή όλες οι ιδιοτιμές του  $I + uu^T$  είναι θετικές (2 και οι  $n-1$  ιδιοτιμές 1), το μητρώο  $I + uu^T$  είναι θετικά ορισμένο (και μάλιστα συμμετρικά θετικά ορισμένο).

**Εξαίρετική προεργασία**

Τώρα όσον αφορά τα ζητούμενα του ερωτήματος:

Έχουμε ότι:  $A_{ij} = I_{ij} + u_i u_j$  και επομένως:

$$A_{ii} = 1 + u_i^2 \quad \text{και} \quad A_{ij} = u_i u_j \quad \text{για} \quad i \neq j$$

Για τη διαγώνιο του L:

$$l_{ii} = \sqrt{1 + u_i^2 - \sum_{k=1}^{i-1} l_{ik}^2}$$

Γιατί; Επίσης χρησιμοποιείς τις τιμές  $l_{ik}$  που δεν τις έχεις ορίσει.

$$\lambda_{jj} = \left( 1 + \frac{u_j^2}{(1 + u_1^2 + \dots + u_{j-1}^2)} \right)^{1/2}$$

$$\lambda_{j+1,j} = \frac{u_{j+1}u_j}{\left( (1 + \sum_{k=1}^{j-1} u_k^2)(1 + \sum_{k=1}^j u_k^2) \right)^{1/2}}$$

Για την υποδιαγώνιο του L:

$$l_{i+1,i} = u_i u_{i+1} \quad \text{για} \quad i \neq j$$

## 2.5 Ερώτημα 5

Έστω ότι  $D = \text{diag}(d_1, \dots, d_n)$ , με  $d_i > 0$  για κάθε  $i$ . Να γράψετε έναν αποδοτικό αλγόριθμο για τον υπολογισμό του μεγαλύτερου στοιχείου του μητρώου  $(D + CC^T)^{-1}$ , όπου  $C \in \mathbb{R}^{n \times r}$ . Υπόδειξη: Να χρησιμοποιήσετε τον τύπο Sherman-Morrison-Woodbury.

### ΑΠΑΝΤΗΣΗ:

Ο τύπος Sherman-Morrison-Woodbury είναι:

$$(B + UV^T)^{-1} = B^{-1} - B^{-1}U(I + V^TB^{-1}U)^{-1}V^TB^{-1}$$

Για το πρόβλημά μας, έχουμε το  $B = D$ ,  $U = C$ , και  $V = C$ , άρα ο τύπος γίνεται:

$$(D + CC^T)^{-1} = D^{-1} - D^{-1}C(I + C^TD^{-1}C)^{-1}C^TD^{-1}$$

Δεδομένου ότι το  $D$  είναι διαγώνιο, μπορούμε να υπολογίσουμε το αντίστροφο του  $D$  εύκολα ως το διαγώνιο μητρώο

$$D^{-1} = \text{diag}\left(\frac{1}{d_1}, \frac{1}{d_2}, \dots, \frac{1}{d_n}\right)$$

Θα μπορούσες να χρησιμοποιήσεις μόνον το διάνυσμα  $\mathbf{d}$ .

Υπολογισμός του  $I + C^T D^{-1} C$ : Αυτό είναι ένα  $r \times r$  μητρώο, το οποίο μπορεί να υπολογιστεί ως το γινόμενο

$$C^T D^{-1} C$$

και προσθέτοντας την μοναδιαία μήτρα  $I$ .

Στη συνέχεια, εφόσον το  $I + C^T D^{-1} C$  είναι μικρότερο μητρώο (διαστάσεων  $r \times r$ ), ο υπολογισμός του αντίστροφου είναι πιο αποδοτικός και μπορεί να γίνει με χρήση της εντολής `inv()` με σημαντικά μικρότερο κόστος σε μνήμη και χρόνο.

Στο αρχείο `.mlx` υπάρχει ο αντίστοιχος αλγόριθμος ο οποίος υλοποιεί τα παραπάνω, καθώς και επιπλέον σύγκριση του χρόνου μεταξύ της άμεσης αντιστροφής με την απευθείας χρήση της `inv()` και της αντιστροφής και εύρεσης του μέγιστου στοιχείου με χρήση της μεθόδου Sherman-Morrison-Woodbury. Παρατηρούμε ότι η δεύτερη μέθοδος είναι σημαντικά γρηγορότερη (για τη σύγκριση των χρόνων έπρεπε κανονικά να μην συμπεριλάβω την εύρεση του μέγιστου στοιχείου αλλά λόγω του ότι πρόκειται για γρήγορη πράξη την συμπεριέλαβα στις μετρήσεις).

## 2.6 Ερώτημα 6

(GvL A4.3.1) Να αναπτύξετε μια εκδοχή του Αλγόριθμου 4.3.1 με την υπόθεση ότι το μητρώο  $A$  αποθηκεύεται σε μορφή ζώνης. (Δείτε την Ενότητα §1.2.5.)

**ΑΠΑΝΤΗΣΗ:** Δες το `.mlx` αρχείο.

## 2.7 Ερώτημα 7

**A4.3.9** Έστω ότι το συμμετρικό και θετικά ορισμένο μητρώο  $A \in \mathbb{R}^{n \times n}$  έχει «δομή βέλους», δηλαδή:

$$A = \begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & 0 & 0 & 0 \\ \times & 0 & \times & 0 & 0 \\ \times & 0 & 0 & \times & 0 \\ \times & 0 & 0 & 0 & \times \end{pmatrix}$$

- (α) Να δείξετε πώς μπορούμε να λύσουμε το γραμμικό σύστημα  $Ax = b$  με  $O(n)$  flop χρησιμοποιώντας τον τύπο Sherman-Morrison-Woodbury.



- (β) Να προσδιορίσετε ένα μητρώο μετάθεσης  $P$  τέτοιο ώστε  $PAP^T = GG^T$ , το οποίο να μπορούμε να υπολογίσουμε με  $O(n)$ .

**ΑΠΑΝΤΗΣΗ:**

**ΑΠΑΝΤΗΣΗ:** α) Προσέξτε ότι το μητρώο μπορεί να γραφτεί ως  $A = D + be_1^T + e_1 c^T$  όπου  $b = [0, D_{2:n,1}^T]^T$  και  $c = [0, D_{1,2:n}]^T$ . Επομένως

$$A = D + [b, e_1][e_1, c]^T$$

(α) ...

και το σύστημα μπορεί να λυθεί με  $\Omega = O(n)$  πράξεις χρησιμοποιώντας τον τύπο SMW.

(β) ...

β) Το μητρώο μετάθεσης είναι το γνωστό  $E_n$ :  $E_n A E_n^T$  είναι το μητρώο σε μορφή "προς τα κάτω βέλους".

## 2.8 Ερώτημα 8

(A4.5.3) Πώς θα λύνατε ένα σύστημα της μορφής:

$$\begin{bmatrix} D_1 & F_1 \\ E_1 & D_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

όπου τα  $D_1$  και  $D_2$  είναι διαγώνια ενώ τα  $F_1$  και  $E_1$  είναι τριδιαγώνια; Υπόδειξη: Να χρησιμοποιήσετε τη μετάθεση τέλειαν αναδιάταξης.

**ΑΠΑΝΤΗΣΗ:** ... **βλ. τέλος παραρτήματος**

# ΕΡΓΑΣΙΑ 3

## Στοιχεία φοιτητή:

Χαράλαμπος Αναστασίου

AM 1093316

4ο Έτος Φοίτησης

## ΕΝΤΟΛΕΣ

### Άσκηση 1

```
n = input('Μέγεθος n μητρώου nxn: ');
N = tril(ones(n), -1);
I = eye(n);
L = I - N;
L_inverse = inv(L)
frob_norm = norm(L_inverse, 'fro');

disp('Frobenius Norm: ')
disp(frob_norm);
```

### Άσκηση 3

```
% Εκτέλεση συνάρτησης po2pp()
po2pp;
```

### Άσκηση 5

```
% Δημιουργία των μητρώων D και C
n = 1000;           % Μέγεθος του διαγώνιου μητρώου D
k = 50;             % Αριθμός στηλών του C (μικρότερος από n)

D = diag(1:n);      % Διαγώνιο μητρώο 1000x1000
C = rand(n, k);      % Τυχαίο μητρώο 1000x50

% Κλήση της συνάρτησης
largest_elem = largest_element_inverse(D, C);

% Εμφάνιση του αποτελέσματος
```

```

disp(['Το μεγαλύτερο στοιχείο του (D + C C^T)^-1 είναι: ',
num2str(largest_elem)]);

% disp(inv(D+C*C'));      % uncomment to check the result

% Μέτρηση του χρόνου εκτέλεσης της largest_element_inverse
time_to_execute_formula = @() largest_element_inverse(D, C);
execution_time_formula = timeit(time_to_execute_formula);
disp(['Ο χρόνος εκτέλεσης της μεθόδου με Sherman-Morrison-Woodbury είναι: ',
num2str(execution_time_formula), ' δευτερόλεπτα']);

% Μέτρηση του χρόνου εκτέλεσης της inv(D + C * C') χωρίς αξιοποίηση της
% μεθόδου S-M-W
time_to_execute_inv = @() inv(D + C * C');
execution_time_inv = timeit(time_to_execute_inv);
disp(['Ο χρόνος εκτέλεσης της inv(D + C * C') είναι: ',
num2str(execution_time_inv), ' δευτερόλεπτα']);

```

## Άσκηση 6

```


A = random_banded_matrix(4, 1, 2)
p=1;
q=2;

% Εκτέλεση της banded LU decomposition
A_band = banded_lu(A, p, q)

[L,U] = lu(A)    % uncomment για σύγκριση με κανονική LU

```

## ΣΥΝΑΡΤΗΣΕΙΣ



```

function packed = po2pp()

% Ορισμός διάστασης του μητρώου από τον χρήστη
n = input('Εισάγετε τη διάσταση του συμμετρικού μητρώου (π.χ. 4 για 4x4): ');

% Αρχικοποίηση του μητρώου
A = zeros(n);

% Εισαγωγή τιμών για το συμμετρικό μητρώο
disp('Εισάγετε τις τιμές του συμμετρικού μητρώου (μόνο για την άνω τριγωνική περιοχή):');

```

```

for i = 1:n
    for j = i:n
        prompt = sprintf('Τιμή για το στοιχείο A(%d, %d): ', i, j);
        A(i, j) = input(prompt);
        A(j, i) = A(i, j);
    end
end

% Επιλογή τρόπου αποθήκευσης (row ή column) από τον χρήστη
storageType = input('Εισάγετε τρόπο αποθήκευσης ("row" για row-major ή
"column" για column-major): ', 's');

% Επιλογή τριγωνικού τμήματος (upper ή lower) από τον χρήστη
trianglePart = input('Εισάγετε τριγωνικό τμήμα ("upper" για άνω ή "lower"
για κάτω): ', 's');

% Αρχικοποίηση μεταβλητής για την αποθηκευμένη μορφή RFP
packed = [];

% Επιλογή για το αν θα αποθηκευτεί το άνω ή το κάτω τριγωνικό τμήμα
if strcmpi(trianglePart, 'upper')
    % Άνω τριγωνικό τμήμα

    if strcmpi(storageType, 'column')
        % Column-major packing
        for j = 1:n
            for i = 1:j
                packed = [packed; A(i, j)];
            end
        end

    elseif strcmpi(storageType, 'row')
        % Row-major packing
        for i = 1:n
            for j = i:n
                packed = [packed; A(i, j)];
            end
        end
    else
        error('Μη έγκυρος τύπος αποθήκευσης. Χρησιμοποιήστε "row" ή
"column".');
    end
elseif strcmpi(trianglePart, 'lower')
    % Κάτω τριγωνικό τμήμα

    if strcmpi(storageType, 'column')
        % Column-major packing
        for j = 1:n
            for i = j:n
                packed = [packed; A(i, j)];
            end
        end

    elseif strcmpi(storageType, 'row')

```

```

        % Row-major packing
        for i = 1:n
            for j = 1:i
                packed = [packed; A(i, j)];
            end
        end
    else
        error('Μη έγκυρος τύπος αποθήκευσης. Χρησιμοποιήστε "row" ή "column".');
    end
else
    error('Μη έγκυρο τριγωνικό τμήμα. Χρησιμοποιήστε "upper" ή "lower".');
end

disp('Το μητρώο στη packed μορφή είναι:');
disp(packed);
end

```

```

function largest_element = largest_element_inverse(D, C)

```

```

    % Υπολογισμός του  $D^{-1}$  (διαγώνιο μητρώο)

```

```

    D_inv = diag(1 ./ diag(D));

```

```

    % Υπολογισμός του  $I + C^T D^{-1} C$ 

```

```

    temp = C' * (D_inv * C);

```

```

    I_plus_temp = eye(size(temp, 1)) + temp;

```

```

    % Υπολογισμός του αντίστροφου του  $I + C^T D^{-1} C$ 

```

```

    I_plus_temp_inv = inv(I_plus_temp);

```

```

    % Υπολογισμός του  $(D + C C^T)^{-1}$  μέσω του τύπου Sherman-Morrison-Woodbury

```

```

    result = D_inv - D_inv * C * (I_plus_temp_inv * (C' * D_inv));

```

```

    % Εύρεση του μεγαλύτερου στοιχείου του αποτελέσματος

```

```

    largest_element = max(max(abs(result)));

```

```

end

```

Προσοχή - το μέγιστο θα είναι στη διαγώνιο (βλ. παράρτημα) επομένως αρκεί να υπολογίσεις τη διαγώνιο και να βρεις το μέγιστο στοιχείο.

```

function A = random_banded_matrix(n, p, q) % Συνάρτηση για δημιουργία τυχαίου
μητρώου ζώνης κάτω εύρους p, άνω εύρους q

```

```

% Δημιουργία μητρώου A μεγέθους nxn με τυχαίες τιμές (φροντίζω να είναι
% ΣΘΟ εδώ - δεν είναι απαραίτητο για την LU)
C = randn(n,n);
A = C'*C;

% Κρατάμε μόνο τα στοιχεία εντός του band με τη χρήση triu και tril
A = tril(triu(A, -p), q);
end

function A_band = banded_lu(A, p, q)
    % A: Πλήρες μητρώο
    % p: Το κάτω εύρος της ζώνης (bandwidth για τον υποδιαγώνιο)
    % q: Το επάνω εύρος της ζώνης (bandwidth για τον υπερδιαγώνιο)

    [n, m] = size(A); % Μέγεθος του μητρώου A

    % Βήμα 1: Αποθήκευση του μητρώου A σε banded μορφή
    A_band = zeros(p + q + 1, n);

    for j = 1:n
        for i = max(1, j - q):min(n, j + p)
            A_band(i - j + q + 1, j) = A(i, j);
        end
    end

    % disp(A_band); % uncomment για να το δω αν βγήκε σωστό

    % Βήμα 2: Banded LU παραγοντοποίηση
    for k = 1:n - 1
        for i = k + 1:min(k + p, n)
            A_band(i - k + q + 1, k) = A_band(i - k + q + 1, k) / A_band(k - k +
q + 1, k);
        end

        for j = k + 1:min(k + q, n)
            for i = k + 1:min(k + p, n)
                A_band(i - k + q + 1, j) = A_band(i - k + q + 1, j) - A_band(i -
k + q + 1, k) * A_band(k - k + q + 1, j);
            end
        end
    end
end
end

```

```
ans = 1x2
    0.3170    0.3170
```

```
error = abs(max(max(inv(diag(d)+C*C'))))
```

```
error =
5.5511e-17
```

```
% max(max(inv(diag(d)+C*C')))
```

```
function [max_B] = find_max_lr
```

```
% Είσοδος: διάνυσμα d και μητρώο C
```

```
[n,r] = size(C); d = d(:);
```

```
% ισοδύναμο με το inv(D)+C*C' (αποδοτικότερο για διαγώνια D)
```

```
F2 = (eye(r) + C'*Cd)\(eye(r) + C'*Cd); % επίλυση n x n
```

```
% υπολογισμός διαγώνιας F2 σύμφωνα με τον τύπο
```

```
for j=1:n, dF(j) = Cd(j,:)*F2(:,j); end
```

```
% υπολογισμός max_B
```

```
max_B = max(dF);
```

```
end
```

## Άσκηση 8

(GvL A4.5.3) Πώς θα λύνατε ένα σύστημα της μορφής  $\begin{pmatrix} D_1 & F_1 \\ E_1 & D_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$  όπου τα  $D_1$  και  $D_2$

είναι διαγώνια ενώ τα  $F_1$  και  $E_1$  είναι τριδιαγώνια; Υπόδειξη: Να χρησιμοποιήσετε τη μετάθεση τέλει αναδιάταξης.

## ΑΠΑΝΤΗΣΗ

Απλοποιούμε θεωρώντας ότι το παραπάνω μητρώο είναι  $2n \times 2n$  και το κάθε μπλοκ  $n \times n$ . Με βάση την υπόδειξη παρατηρούμε ότι αν  $P_{2,n}$  είναι το μητρώο τέλει αναδιάταξης, τότε  $P_{2,n}AP_{2,n}^T$  μπορεί να θεωρηθεί ως μπλοκ  $2 \times 2$  όπου τα διαγώνια μπλοκ είναι ζώνης και τα εξωδιαγώνια μπλοκ είναι πολύ χαμηλής τάξης. Ειδικότερα: Τα διαγώνια μπλοκ έχουν κάτω και άνω εύρος ζώνης 3, επομένως είναι 7-διαγώνια. Τα εξωδιαγώνια μπλοκ είναι  $B_{21} = \gamma_1 e_1 e_n^T + \gamma_2 e_2 e_{n-1}^T$  και  $B_{12} = \delta_1 e_n e_1^T + \delta_2 e_{n-1} e_2^T$ . Προσέξτε ότι το ανάστροφο του  $B_{21}$  έχει τη δομή του  $B_{12}$ .

Δεδομένων των παραπάνω: Υποθέτοντας ότι τα διαγώνια μπλοκ  $B_{11}, B_{22}$  είναι αντιστρέψιμα, ακολουθούμε την στρατηγική Spike.

Μία άλλη ιδέα είναι να εφαρμόσουμε μπλοκ LU στο μητρώο ώστε να προκύψουν οι παράγοντες:

$$\begin{pmatrix} D_1 & F_1 \\ E_1 & D_2 \end{pmatrix} = \begin{pmatrix} I & 0 \\ E_1 D_1^{-1} & I \end{pmatrix} \begin{pmatrix} D_1 & F_1 \\ 0 & D_2 - E_1 D_1^{-1} F_1 \end{pmatrix}$$

Στη συνέχεια η επίλυση μπορεί να γίνει βάσει ενός βήματος μπλοκ εμπρός αντικατάστασης ενός βήματος μπλοκ πίσω αντικατάστασης. Ειδικότερα, δεδομένου ότι:

$$\begin{pmatrix} I & 0 \\ E_1 D_1^{-1} & I \end{pmatrix} \begin{pmatrix} D_1 & F_1 \\ 0 & D_2 - E_1 D_1^{-1} F_1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Βήμα 1:  $y_1 = b_1$ ,  $y_2 = b_2 - E_1 D_1^{-1} y_1$

Βήμα 2:  $(D_2 - E_1 D_1^{-1} F_1) x_2 = y_2$ ,  $x_1 = D_1^{-1} (y_1 - F_1 x_2)$

Δεδομένης της δομής των μπλοκ (διαγώνια  $D_i$ , τριδιαγώνια  $E_1, F_1$ ), το βασικό κόστος στα παραπάνω οφείλεται στην επίλυση με το συμπλήρωμα Schur (Βήμα 2, υπολογισμός του  $x_2$ ). Το μητρώο είναι 5-διαγώνιο (αποδεικνύεται καθώς το γινόμενο δύο τριδιαγώνιων μητρώων, δηλ.  $E(1|1)F(1|1)$  είναι  $C(2|2)$ , επομένως σχετικά μικρής ζώνης, και λύνεται με  $\Omega = O(n)$ . Προσοχή: Η ευσταθής υλοποίηση του παραπάνω προϋποθέτει την αντιστρεψιμότητα των  $D_i$  (δηλ. να μην περιέχουν μηδενικά στοιχεία στη διαγώνιό τους) και την καλή κατάσταση του συμπληρώματος Schur  $S = (D_2 - E_1 D_1^{-1} F_1)$ . Αυτά δεν είναι εξασφαλισμένα! Π.χ. θα μπορούσε να ισχύει  $D_1 = D_2 = 0$  και το μητρώο να είναι αντιστρέψιμο (δοκιμάστε π.χ.  $E_1 = F_1 = I_4$ ) οπότε ο παραπάνω αλγόριθμος δεν μπορεί να εφαρμοστεί ως έχει.

```
n=10;
d1 = diag(8*eye(n)); d2 = d1/2;
E = toeplitz([4,-1,zeros(1,n-2)]); F=toeplitz([2,-1,zeros(1,n-2)]);
A = [diag(d1),F;E,diag(d2)]; b = A*ones(2*n,1);
psh = shuffle_perm(2,10)
spy(A(psh,psh))

xlim([0.0 21.0])
ylim([0.0 21.0])
[x] = block_trid0(d1,d2,E,F,b);
f = @( ) block_trid0(d1,d2,E,F,b);
%timeit(f)
f_simple = @( ) A\b;
%timeit(f_simple)
norm(A\b-x)
x_dlu = block_trid1(d1,d2,E,F,b);
norm(A\b-x_dlu)
```

Η συνάρτηση είναι:

```
function [x] = block_trid0(d1,d2,E,F,b);
% επίλυση του συστήματος [diag(d1),F;E,diag(d2)]x=b
% όπου D διαγώνια και E, F τριδιαγώνια
[m1,n1]=size(E); if (m1~=length(d2))|(n1~=length(d1)) error('ασυνεπή μεγέθη');
end
[m2,n2]=size(F); if (m2~=length(d1))|(n2~=length(d2)) error('ασυνεπή μεγέθη');
end
if (~isbanded(E,1,1)|~isbanded(F,1,1)) error('τα εξωδιαγώνια μητρώα πρέπει να είναι τριδιαγώνια'); end
b1 = b(1:m2); b2 = b(m2+1:end);
% Βήμα 1
y1 = b1; y2 = b2 - E*(y1./d1);
% Βήμα 2
x2 = (diag(d2)-E*(diag(d1)\F))\y2;
x1 = (y1-F*x2)./d1;
x = [x1;x2];
end
```



Εναλλακτική πρόταση:

Γενικεύουμε για μπλοκ την ιδέα που είδαμε για την επίλυση τριδιαγώνιων μητρώων  $T = D - L - U$  (D διαγώνιο, L (U) αυστηρά κάτω (άνω) διδιαγώνιο) με διαδοχικούς μετασχηματισμούς, π.χ. στο 1ο βήμα γράφοντας  $Tx = b \Rightarrow (D + L + U)D^{-1}Tx = (D + L + U)D^{-1}b$  κ.ο.κ. Στην περίπτωση μας έχουμε  $D = [D_1, 0; 0, D_2]$  επομένως η διαδικασία έχει ως εξής:

$$\begin{pmatrix} D_1 & -F \\ -E & D_2 \end{pmatrix} \begin{pmatrix} D_1^{-1} & 0 \\ 0 & D_2^{-1} \end{pmatrix} \begin{pmatrix} D_1 & F \\ E & D_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} D_1 & -F \\ -E & D_2 \end{pmatrix} \begin{pmatrix} D_1^{-1} & 0 \\ 0 & D_2^{-1} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

και εκτελώντας τις πράξεις και απλοποιήσεις προκύπτει το μπλοκ διαγώνιο σύστημα:

$$\begin{pmatrix} D_1 - F D_2^{-1} E & 0 \\ 0 & D_2 - E D_1^{-1} F \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 - F D_2^{-1} b_2 \\ b_2 - E D_1^{-1} b_1 \end{pmatrix}$$

Αν και το κόστος φαίνεται μεγαλύτερο από τον προηγούμενο αλγόριθμο, το ενδιαφέρον είναι ότι οι πράξεις για τον υπολογισμό των  $x_1$  και  $x_2$  μπορούν να εκτελεστούν ανεξάρτητα (αν και κάθε επεξεργαστής πρέπει να έχει πρόσβαση στα δεδομένα που απαιτούνται για τον υπολογισμό) και επομένως ταυτόχρονα σε ένα παράλληλο σύστημα. Επίσης ότι αν για παράδειγμα χρειάζεται μόνον το  $x_1$  ή το  $x_2$ , δεν χρειάζεται ο έτερος.

Παραθέτουμε παρακάτω συνάρτηση για την επίλυση βασισμένη στα παραπάνω:

```
function [x] = block_trid1(d1,d2,E,F,b);
% επίλυση του συστήματος [diag(d1),F;E,diag(d2)]x=b
% όπου D διαγώνια και E, F τριδιαγώνια
[m1,n1]=size(E); if (m1~=length(d2))|(n1~=length(d1)) error('ασυνεπή μεγέθη');
end
[m2,n2]=size(F); if (m2~=length(d1))|(n2~=length(d2)) error('ασυνεπή μεγέθη');
end
if (~isbanded(E,1,1)|~isbanded(F,1,1)) error('τα εξωδιαγώνια μητρώα πρέπει να είναι τριδιαγώνια'); end
b1 = b(1:m2); b2 = b(m2+1:end);
% Βήμα 1
y1 = b1-F*(b2./d2); y2 = b2 - E*(b1./d1);
% Βήμα 2
x2 = (diag(d2)-E*(diag(d1)\F))\y2;
x1 = (diag(d1)-F*(diag(d2)\E))\y1;
x = [x1;x2];
end
```