Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής

Πανεπιστήμιο Πατρών

Εργαστηριακή Άσκηση
Επιστημονικός Υπολογισμός
CEID 1151

Ανδρέας Καρατζάς

22 Φεβρουαρίου 2021

Περιεχόμενα

1. Εισαγωγικά

Υποερώτημα 1.1

Στοιχεία υπολογιστικού συστήματος

Χαρακτηριστικό	Απάντηση
Έναρξη / λήξη εργασίας	31/1/2021 - 22/2/2021
model	OMEN by HP Laptop 15-dc1xxx[1]
O/S	Microsoft Windows 10 Education 10.0.19042[2]
processor name	Intel Core i7 9750H[3]
processor speed	2.6 GHz (base)[4]
number of processors	1[5]
total # cores	6[6]
total # threads	12[7]
FMA instruction	yes[8]
L1 cache	192 KB Instruction, 192 KB Data write-back[9]
L2 cache	(per core) 256 KB, write-back[10]
L3 cache	(shared) 12 MB, write-back[11]
Gflops/s	337.4[12]
Memory	16 GB[13]
Memory Bandwidth	41.8 GB/s[14]
MATLAB Version	9.7.0.1434023 (R2019b) Update 6[15]
BLAS	Intel(R) Math Kernel Library Version 2018.0.3 Product Build 20180406 for Intel(R) 64 architecture applications, CNR branch AVX2[16]
LAPACK	Intel(R) Math Kernel Library Version 2018.0.3 Product Build 20180406 for Intel(R) 64 architecture applications, CNR branch AVX2 Linear Algebra PACKage Version 3.7.0[17]

Πίνακας 1.1: Στοιχεία για τα πειράματα

Εκτελώντας την εντολή `bench` στο περιβάλλον *MATLAB* προκύπτουν τα αποτελέσματα που φαίνονται στο σχήμα 1.1.

Computer Type	LU	FFT	ODE	Sparse	2-D	3-D
This machine	0.0796	0.0606	0.0133	0.0801	0.3929	0.3384
Windows 7, Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50 GHz	0.0767	0.0979	0.0154	0.1007	0.2420	0.2746
Linux 18.04, Intel Xeon CPU E5-2665 0 @ 2.40 GHz	0.0766	0.0969	0.0147	0.1126	0.3538	0.2652
Windows 10, Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50 GHz	0.0766	0.0969	0.0147	0.1126	0.3538	0.2652
Windows 10, Intel(R) Xeon(R) W-2133 @ 3.60 GHz	0.0814	0.0870	0.0138	0.1185	0.3117	0.3877
iMac, macOS 10.13.6, Intel Core i7 @3.4 GHz	0.1501	0.1354	0.0250	0.1181	0.4019	0.3264
Windows 10, AMD Ryzen 7 1700 @ 3.00 GHz	0.1840	0.1441	0.0158	0.2030	0.2959	0.3000
Surface Pro 3, Windows 10, Intel Core i5-4300U @ 1.9 GHz	0.1957	0.1764	0.0227	0.1391	0.6722	0.4643
MacBook Pro, macOS 10.14.1, Intel Core i5 @ 2.6 GHz	0.2734	0.1980	0.0177	0.1423	2.0198	1.3889

Σχήμα 1.1: Ο πίνακας με τα αποτελέσματα της εντολής `bench` στο περιβάλλον *MATLAB*

Υποερώτημα 1.2

Δομή του συμπιεσμένου

Στο συμπιεσμένο αρχείο βρίσκονται:

- Η αναφορά της εργασίας, με όνομα αρχείου `2016_1054336_Karatzas.pdf`
- Ο φάκελος `question_2` με όλα τα εκτελέσιμα αρχεία **MATLAB**, όπως επίσης και τα αραιά μητρώα, που αφορούν το ερώτημα 2
- Ο φάκελος `question_3` με όλα τα εκτελέσιμα αρχεία **MATLAB** που αφορούν το ερώτημα 3
- Ο φάκελος `question_4` με όλα τα εκτελέσιμα αρχεία **MATLAB**, όπως επίσης και τα αραιά μητρώα, που αφορούν το ερώτημα 4

[1] Για την ανάκτηση του ονόματος του μοντέλου, εκτελέστηκε η εντολή `Get-ComputerInfo` σε περιβάλλον *Powershell* v5.1.
[2] Για την ανάκτηση του ονόματος του λειτουργικού καθώς και της έκδοσής του, εκτελέστηκε η εντολή `Get-ComputerInfo` σε περιβάλλον *Powershell* v5.1.
[3] Για την ανάκτηση του ονόματος του επεξεργαστή, εκτελέστηκε η εντολή `Get-ComputerInfo` σε περιβάλλον *Powershell* v5.1.
[4] Δείτε την επίσημη ιστοσελίδα της Intel.
[5] Δείτε την επίσημη ιστοσελίδα της Intel.
[6] Δείτε την επίσημη ιστοσελίδα της Intel.
[7] Δείτε την επίσημη ιστοσελίδα της Intel.
[8] Δείτε το παράρτημα στη σελίδα wikichip για το συγκεκριμένο επεξεργαστή.
[9] Δείτε το παράρτημα στη σελίδα wikichip για το συγκεκριμένο επεξεργαστή.
[10] Δείτε το παράρτημα στη σελίδα wikichip για το συγκεκριμένο επεξεργαστή.
[11] Δείτε το παράρτημα στη σελίδα wikichip για το συγκεκριμένο επεξεργαστή.
[12] Δείτε το παράρτημα στη σελίδα gadgetversus για το συγκεκριμένο επεξεργαστή.
[13] Για την ανάκτηση του μεγέθους της μνήμης, εκτελέστηκε η εντολή `Get-CimInstance -Class CIM_PhysicalMemory -ErrorAction Stop | Select-Object *` σε περιβάλλον *Powershell* v5.1.
[14] Δείτε την επίσημη ιστοσελίδα της Intel.
[15] Εκτελέστηκε η εντολή `version` στο περιβάλλον *MATLAB*.
[16] Εκτελέστηκε η εντολή `version -blas` στο περιβάλλον *MATLAB*.
[17] Εκτελέστηκε η εντολή `version -lapack` στο περιβάλλον *MATLAB*.

- Ο φάκελος `question_5` με όλα τα εκτελέσιμα αρχεία **MATLAB** που αφορούν το ερώτημα 5

Για την εκτέλεση των *scripts* της εργασίας, θα πρέπει να έχουν προστεθεί στο *path* της **MATLAB** τα πακέτα:

- `ssget`
- Tensor Toolbox for MATLAB, Version 3.2

Τα παραπάνω πακέτα δεν υπάρχουν στο συμπιεσμένο.

2. Αραιή αναπαράσταση BCRS

Το πρόβλημα της αποθήκευσης αραιών μητρώων απασχολεί την επιστημονική κοινότητα που σχετίζεται με προβλήματα επιστημονικού υπολογισμού πολύ καιρό [4]. Ο στόχος είναι η αποδοτική αναπαράσταση των μητρώων με άξονα τη μνήμη. Το πρώτο μοντέλο αναπαράστασης αραιών μητρώων ήταν το COO που κατάφερε να μειώσει σημαντικά το μέγεθος ενός αραιού μητρώου στη μνήμη. Ωστόσο, παρατηρήθηκε πως και το COO μπορούσε να βελτιωθεί. Έτσι, ανακαλύφθηκαν οι μέθοδοι CSR και CSC οι οποίες χρησιμοποιώντας *indexing*. Μείωσαν ακόμα παραπάνω το μέγεθος μνήμης που καταλάμβανε ένα αραιό μητρώο στη μνήμη. Το πρόβλημα που προέκυψε όμως στην πράξη ήταν ότι αυτές οι μέθοδοι δεν εκμεταλλεύονταν την τοπικότητα στη μνήμη με αποτέλεσμα να υπάρχουν αρκετά cache misses και οι εφαρμογές να είναι *memory bound*. Έτσι, άρχισαν να ανακαλύπτονται *block* μέθοδοι για επιστημονικούς υπολογισμούς. Φορτώνοντας *block-by-block* κι όχι *element-by-element* τα δεδομένα του αραιού μητρώου, αυξάνεται το πλήθος των μηδενικών ("άχρηστων") στοιχείων που πρέπει να αποθηκευτούν, αλλά μειώνεται ο συνολικός αριθμός των αποτυχιών της κρυφής μνήμης (cache misses).

Υποερώτημα 2.1

Η Συνάρτηση `sp_mx2bcrs`

Η συνάρτηση `sp_mx2bcrs` έχει υλοποιηθεί, όπως ορίζεται από την εκφώνηση. Η συνάρτηση καλεί κατά τη διάρκεια της εκτέλεσης την εξωτερική συνάρτηση `nnz_blk` η οποία επιστρέφει ένα δυαδικό μητρώο που έχει 1 σε κάθε στοιχείο που αντιστοιχεί σε μη μηδενικό block του μητρώου A και 0 σε όλα τα μηδενικά blocks. Επομένως:

$$\text{Αν } A = \begin{bmatrix} 0 & 0.4548 & 0 & 0 & 0 & 0 \\ 0 & 0.7749 & 0 & 0.0349 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.3755 & 0 \\ 0.7641 & 0 & 0.4764 & 0.5153 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1538 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ και } block\ size = 2,$$

$$\text{τότε } B = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Ο αλγόριθμος έχει πολυπλοκότητα $O(2 \cdot n)$, καθώς χρειάζεται n προσπελάσεις των του μητρώου A για την αρχικοποίηση του μητρώου B, και n προσπελάσεις για την αναπαράσταση σε *Block Compressed Row Storage*. Ωστόσο, επειδή αυτή η συνάρτηση προορίζεται για αραιά μητρώα, η μέση πολυπλοκότητα είναι πολύ μικρότερη και μάλιστα μικραίνει ανάλογα με το πόσο αραιό είναι το μητρώο και πόσο κατάλληλο είναι το μέγεθος *block* που επιλέχθηκε για την αναπαράστασή του[1].

[1] Το *block size* έχει να κάνει συνήθως με το μέγεθος της *cache* του εκάστοτε συστήματος, αλλά μπορεί να βρεθεί κι εκεί μια ισορροπία μεταξύ εκμετάλλευσης της τοπικότητας και της πολυπλοκότητας αναπαράστασης του μητρώου σε *BCRS* μορφή.

Υποερώτημα 2.2

Η Συνάρτηση `spmv_bcrs`

Η συνάρτηση `spmv_bcrs` έχει υλοποιηθεί, όπως ορίζεται από την εκφώνηση. Η πολυπλοκότητα της πράξης $y \leftarrow y + A \cdot x$ (*GEMV*) όταν το μητρώο είναι αραιό και αποθηκευμένο σε *BCRS* μορφή είναι $O(m)$, όπου m είναι ο αριθμός των μη μηδενικών *blocks*. Έτσι και η συνάρτηση που υλοποιήθηκε στα πλαίσια της εργασίας πετυχαίνει αυτήν την πολυπλοκότητα.

Υποερώτημα 2.3

Επικύρωση Συναρτήσεων

Για την επικύρωση των συναρτήσεων `sp_mx2bcrs` και `spmv_bcrs` υλοποιήθηκε το *script* `validate_bcrs_gemv.m`. Για κάθε πείραμα, το *script*:

1. Φορτώνει το αραιό μητρώο
2. Επιλέγει τυχαία ένα *block size*
3. Καλεί τη συνάρτηση `sp_mx2bcrs`
4. Δημιουργεί 2 ψευδοτυχαία διανύσματα x και y
5. Καλεί τη συνάρτηση `spmv_bcrs`
6. Εκτελεί τη πράξη $y \leftarrow y + A \cdot x$ χρησιμοποιώντας τους τελεστές της **MATLAB**
7. Βρίσκει τη νόρμα *Frobenius* της διαφοράς του αποτελέσματος του 5ου βήματος από του 6ου βήματος
8. Αποθηκεύει τα αποτελέσματα του πειράματος σε μια μεταβλητή τύπου *table*

Για την αρχικοποίηση του *block size* στο κάθε πείραμα δημιουργήθηκε η συνάρτηση `get_divisors` η οποία επιστρέφει όλους τους διαιρέτες του μεγέθους του εκάστοτε μητρώου. Από αυτούς τους διαιρέτες επιλέγεται τυχαία κάποιος αριθμός που θα λειτουργήσει ως *block size*. Τα μητρώα που επιλέχθηκαν από τη SuiteSparse βρίσκονται στον πίνακα 2.1.

ID	Name	Rows	Columns	Nonzeros	Εντολή `ssget`
1896	circuit204	1,020	1,020	5,883	`ssget('YZhou/circuit204')`
1638	tols2000	2,000	2,000	5,184	`ssget('Bai/tols2000')`

Πίνακας 2.1: Στοιχεία αραιών μητρώων για την επικύρωση των συναρτήσεων `sp_mx2bcrs` και `spmv_bcrs`

Τα μητρώα επιλέχθηκαν έτσι ώστε να έχουν ένα ικανοποιητικό αριθμό μη μηδενικών στοιχείων και να αποτελούν καλά διανύσματα[2] ελέγχου. Για παράδειγμα, ένα αραιό μητρώο διαστάσεων $n \times n$ το οποίο έχει n μη μηδενικά στοιχεία, είναι πολύ πιθανό να είναι ένα διαγώνιο μητρώο. Επομένως, κύριο μέλημα ήταν τα μητρώα να μην εμφανίζουν κάποιο δομικό χαρακτηριστικό, αλλά να είναι γενικά, αραιά και τετραγωνικά μητρώα.

[2] Ο όρος διάνυσμα αναφέρεται στη συγκεκριμένη περίπτωση στα διανύσματα ελέγχου για την επικύρωση των συναρτήσεων και όχι στη μαθηματική οντότητα.

3. Τανυστές και διαδρομές

Το συγκεκριμένο ερώτημα προκάλεσε ιδιαίτερο ενδιαφέρον. Αρχικά, εξετάστηκε η περίπτωση χρήσης κάποιου υπερταχούς αλγορίθμου για πολλαπλασιασμό μητρώων, όπως ο αλγόριθμος *Strassen* ή και ο αλγόριθμος *Coppersmith-Winnograd*. Σε αυτό το ερώτημα έγινε αρκετή έρευνα για το πόσο πρακτικοί είναι αυτοί οι αλγόριθμοι. Βρέθηκε πως ο αλγόριθμος *Coppersmith-Winnograd*, που έχει και τη μικρότερη πολυπλοκότητα από τους 2, δεν χρησιμοποιείται πρακτικά καθώς τα πλεονεκτήματα αρχίζουν μόνο για πολύ μεγάλα μητρώα, μητρώα που δεν μπορεί το παρών υλικό να "επεξεργαστεί"[1]. Το επόμενο βήμα ήταν να χρησιμοποιηθεί κάποιος υπερταχής αλγόριθμος πολλαπλασιασμού ακέραιων αριθμών, όπως ο αλγόριθμος Karatsuba. Ο αλγόριθμος *Karatsuba* υπάρχει υλοποιημένος σε **MATLAB** και παρουσιάζεται στον κώδικα 3.1.

Listing 3.1: Ο αλγόριθμος *Karatsuba*

```
% This was downloaded from: https://www.mathworks.com/matlabcentral/fileexchange/
% 73060-karatsuba-algorithm-for-fast-multiplication

% Multiplication of "x" and "y" with Karatsuba method using base "base"
% x , y and base can be freely chosen
function xy = karatsuba(x, y, base)
    if (x <= base && y <= base) || x == 0 || y == 0
        xy = x .* y;
        return;
    else
        % find smallest m with 2^m >= max(noDigits(x,base), noDigits(y,base))
        m = ceil(log2(max(noDigits(x, base), noDigits(y, base))));
        % split x and y in two pieces (xl|xr) and (yl|yr)
        splitpoint = base.^(2^m/2);
        xl = floor(x ./ splitpoint);
        xr = x - xl.*splitpoint;
        yl = floor(y ./ splitpoint);
        yr = y - yl.*splitpoint;

        zl = karatsuba(xl,yl,base);
        zr = karatsuba(xr,yr,base);
        zmiddle = karatsuba( xl+xr , yl+yr, base);
        xy = splitpoint^2.*zl +  splitpoint.*(zmiddle - zl - zr) + zr;
    end
end

% returns the number of digits of n using base "base"
% example: 31 is a 2 digit number with base = 10 and a 5 digit
% number with base = 2
function d = noDigits(n,base)
    d = floor(log10(n) / log10(base)) + 1;
end
```

Το πρόβλημα ήταν ότι οι περισσότερες συναρτήσεις της **MATLAB** δε δέχονται ορίσματα τύπου `int` και θα χρειάζονταν πολλές μετατροπές μεταβλητών (*type casting*). Ωστόσο, ο αλγόριθμος δοκιμάστηκε και βρέθηκε ότι λόγω των βελτιστοποιήσεων που έχουν γίνει στη **MATLAB**[2] είχε χειρότερα αποτελέσματα από τον τελεστή πολλαπλασιασμού της **MATLAB**. Επομένως, χρησιμοποιήθηκε ο τελεστής πολλαπλασιασμού της **MATLAB**.

Τέλος, αναφέρεται πως για την εκτέλεση των συναρτήσεων θα πρέπει να έχει γίνει λήψη του Tensor Toolbox for MATLAB, Version 3.2, όπως αναφέρεται και στην εκφώνηση.

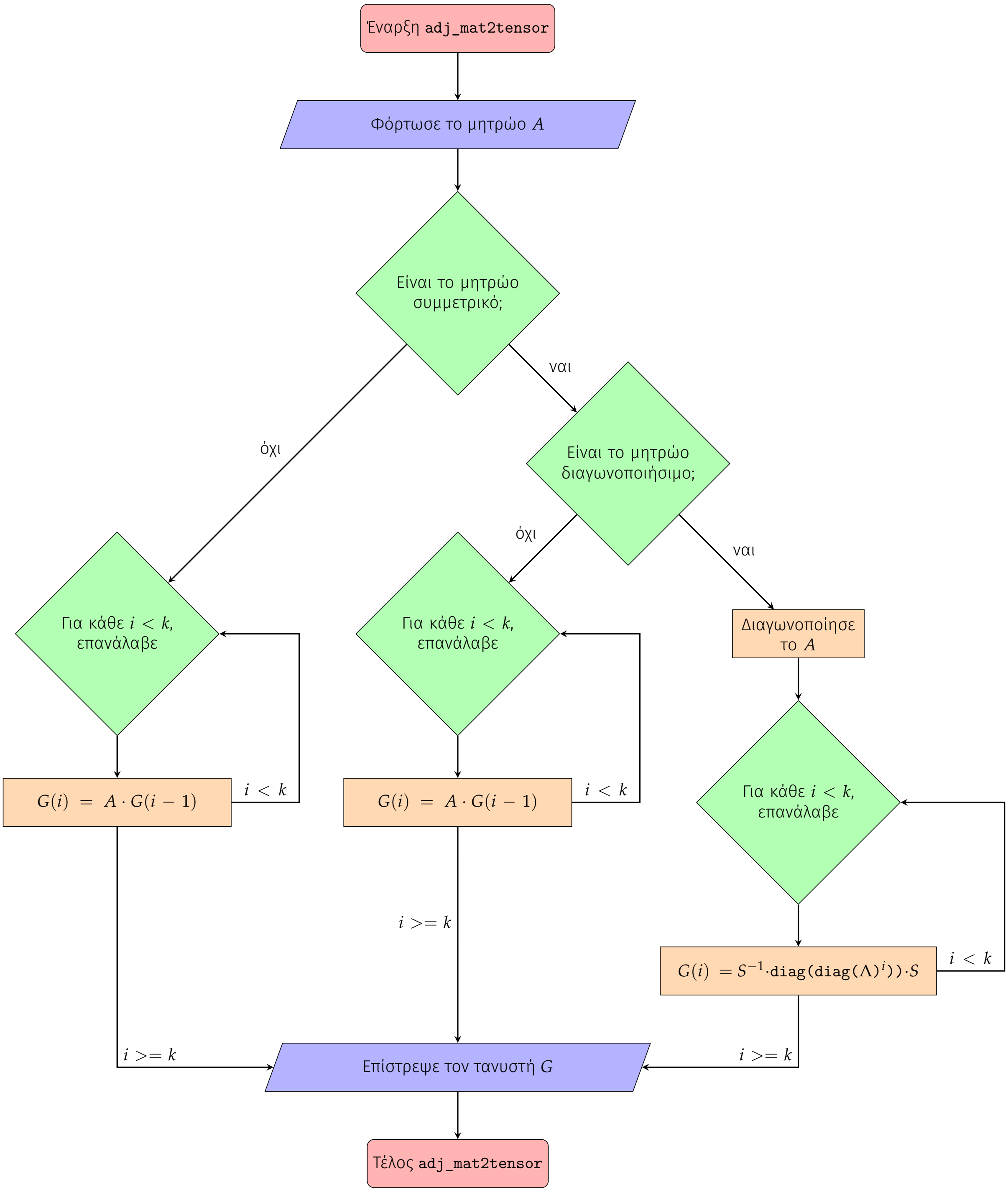
[1] `https://mathoverflow.net/questions/101531/how-fast-can-we-really-multiply-matrices`

[2] Η **MATLAB** χρησιμοποιεί MKL με αποτέλεσμα να εκμεταλλεύεται συγκεκριμένα χαρακτηριστικά του υλικού του ξενιστή (host) - συστήματος.

Υποερώτημα 3.1

Η συνάρτηση `adj_mat2tensor`

Για τη δημιουργία του τανυστή G δημιουργήθηκε η συνάρτηση **`adj_mat2tensor`**, όπως περιγράφεται από την εκφώνηση. Η συνάρτηση εκμεταλλεύεται τα διαγωνοποιήσιμα μητρώα για τα οποία υπάρχει μια συντόμευση για τον υπολογισμό των δυνάμεων του μητρώου[3]. Το *flow chart* της συνάρτησης δίνεται στο σχήμα 3.1.



Σχήμα 3.1: Το *flow chart* της **`adj_mat2tensor`**

[3]Βλέπε Εισαγωγή στη Γραμμική Άλγεβρα, του Gilbert Strang, Κεφάλαιο 6 Παράγραφο 2 [6].

Αρχικά, το πλήθος διαδρομών μήκους ως k σε ένα γράφημα G δεδομένου του αντίστοιχου μητρώου γειτνίασης A δίνεται από την έκφραση A^k [5]. Υπάρχουν 2 τρόποι για τον υπολογισμό της έκφρασης A^k, ανάλογα με τη δομή και τα χαρακτηριστικά του μητρώου. Αν το μητρώο είναι διαγωνοποιήσιμο, η πράξη A^k δίνεται από την έκφραση $S^{-1} \cdot \Lambda^k \cdot S$. Επειδή όμως το Λ είναι διαγώνιο μητρώο, η πράξη Λ^k είναι ουσιαστικά πολλαπλασιασμός διανυσμάτων κι όχι μητρώων.

Αυτή η συντόμευση είναι αρκετά χρήσιμη σε περιπτώσεις όπου τα περισσότερα γραφήματα είναι μη-κατευθυνόμενα. Σε ένα μη κατευθυνόμενο γράφημα, το μητρώο γειτνίασης είναι συμμετρικό. Χρησιμοποιώντας την τεχνική της διαγωνοποίησης το κόστος της πράξης A^i, $i \in \Re$ μειώνεται από $O(GEMM) \cdot (i-1) = n^2 \cdot (2 \cdot n - 1) \cdot (i-1)$ σε $n^2 \cdot (i-1) + O(\texttt{eig()})$, μειώνοντας κατά μία τάξη μεγέθους την πολυπλοκότητα της ζητούμενης πράξης. Ωστόσο, αυτό θα φανεί πραγματικά χρήσιμο σε στατικά γραφήματα κι όχι σε δυναμικά. Στα δυναμικά γραφήματα, όπου υπάρχουν συνεχείς εισαγωγές και διαγραφές κόμβων, οι ιδιοτιμές και τα ιδιοδιανύσματα του μητρώου γειτνίασης του γραφήματος αλλάζουν συνεχώς και άρα κάθε φορά θα πρέπει να προστίθεται το κόστος της πράξης `eig()`.

Η συνάρτηση `adj_mat2tensor` καλεί τη συνάρτηση `is_defective` για να εντοπισθεί τυχόν μη διαγωνοποιήσιμο μητρώο. Επίσης, για τη δοκιμή (*test*) της συνάρτησης `adj_mat2tensor` δημιουργήθηκε και η συνάρτηση `gen_adj_mat`, η οποία δημιουργεί ψευδοτυχαία μητρώα γειτνίασης είτε για την περίπτωση μη κατευθυνόμενου γραφήματος ($\texttt{sym\_flag} = 1$), είτε για την περίπτωση κατευθυνόμενου γραφήματος ($\texttt{sym\_flag} = 0$). Θεωρείται πως τα μητρώα γειτνίασης ακολουθούν τον ορισμό:

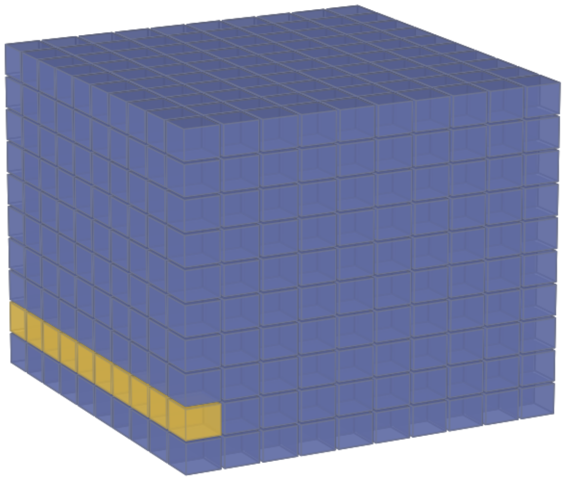
$$A_{uv} = \begin{cases} 1, & \text{if } uv \in E \\ 0, & \text{if } uv \notin E \end{cases},$$

όπου $G = \{V, E\}$ το εκάστοτε γράφημα με το σύνολο των ακμών του G. Τέλος, ο εξεταστής μπορεί να εκτελέσει το *script* `test_adj_mat2tensor`, με το οποίο μπορεί να αξιολογήσει την ορθότητα των προαναφερθέντων συναρτήσεων. Ο εξεταστής θα πρέπει να προσέξει να μην αλλάξει τη μεταβλητή k υπερβολικά, προκαλώντας υπερχείλιση.

Υποερώτημα 3.2

Η συνάρτηση `get_ten_fiber`

Χρησιμοποιώντας τον τανυστή που επιστρέφει η συνάρτηση `adj_mat2tensor` ζητείται από το παρόν υποερώτημα ο υπολογισμός του πλήθους των διαδρομών μήκους ως k μεταξύ 2 κόμβων (i, j). Ο τανυστής περιέχει πληροφορία για το πλήθος των διαδρομών μεταξύ οποιωνδήποτε 2 κόμβων του γραφήματος (i, j) μήκους έως k. Στο 1ο *slice*, ο τανυστής G είναι ίσος με το A. Στο l-οστό *slice*, ο τανυστής G περιέχει πληροφορία σχετικά με το πλήθος των διαδρομών έως l μεταξύ οποιουδήποτε ζευγαριού κόμβων (i, j). Επομένως, για τον υπολογισμό του πλήθους διαδρομών μήκους έως k μεταξύ 2 δεδομένων κόμβων (i, j), θα πρέπει να υπολογισθεί το άθροισμα $G(i,j,l)$, $\forall l \in [1, k]$. Αυτό το άθροισμα είναι ισοδύναμο με το άθροισμα των στοιχείων της *ίνας* $G(i,j,:)$. Για παράδειγμα, η ίνα για το ζευγάρι κόμβων $(9, 1) = G(9,1,:)$ φαίνεται στο σχήμα 3.2 και περιέχει το πλήθος διαδρομών μεταξύ μήκους l έως 10.



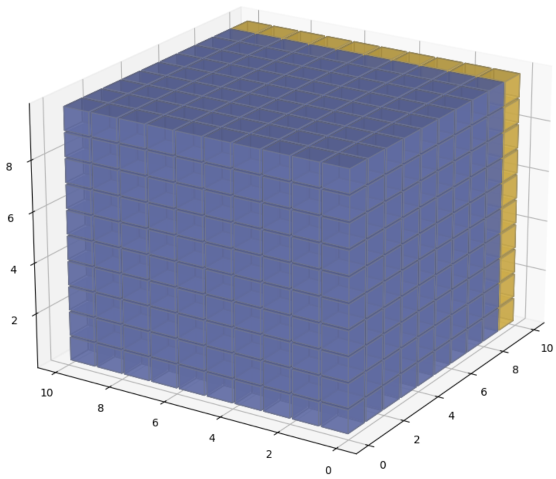
Σχήμα 3.2: Η ίνα $G(9,1,:)$, για ένα τανυστή $G^{10\times10\times10}$

Αθροίζοντας τα στοιχεία της εκάστοτε ίνας, υπολογίζεται το ζητούμενο σύνολο. Η διαδικασία αυτή υπάρχει με κώδικα στη συνάρτηση `get_ten_fiber`.

Υποερώτημα 3.3

Η συνάρτηση `collapse_ten`

Χρησιμοποιώντας και πάλι τον τανυστή που επιστρέφει η συνάρτηση `adj_mat2tensor` ζητείται από το παρόν υποερώτημα ο υπολογισμός του πλήθους των διαδρομών μήκους ως k μεταξύ όλων των ζευγαριών κόμβων (i, j). Αυτό δε διαφέρει πολύ από το προηγούμενο υποερώτημα (3.2). Ουσιαστικά εδώ θα πρέπει να αθροιστούν όλες οι "μπροστινές" φλύδες[4] (*frontal slices*) μεταξύ τους και να επιστραφεί έτσι το ζητούμενο μητρώο. Στο σχήμα 3.3 φαίνεται ένα παράδειγμα "μπροστινής" φλύδας.



Σχήμα 3.3: Η φλύδα $G(:,:,10)$, για ένα τανυστή $G^{10\times10\times10}$

[4]Για τον όρο *frontal slices* δε βρέθηκε κάποια ελληνική μετάφραση.

Με άξονα το παράδειγμα του σχήματος 3.3, η λειτουργία που ζητείται στο συγκεκριμένο υποερώτημα είναι ο υπολογισμός του αθροίσματος με *element-wise* τρόπο όλων των *frontal slices* του τανυστή G, δηλαδή το άθροισμα των στοιχείων των 10 *frontal slices*. Αυτό υλοποιείται στη συνάρτηση `collapse_ten`.

4. Στατιστικά μητρώων

Για το συγκεκριμένο ερώτημα, δόθηκαν διευκρινήσεις μέσω ηλεκτρονικού ταχυδρομείου. Με βάση αυτές τις διευκρινήσεις:

1. $k = 1, 3, \cdots, 2 \cdot p - 1$
2. Το k είναι το *bandwidth* του μητρώου

Επίσης, για την εκτέλεση των συναρτήσεων θα πρέπει να υπάρχει η συνάρτηση ssget στο *PATH* της **MATLAB**, έτσι ώστε να γίνει λήψη των μητρώων που ζητείται στα πλαίσια του ερωτήματος από τη SuiteSparse.

Υποερώτημα 4.1

Η συνάρτηση `band_stats`

Η συνάρτηση `band_stats` έχει υλοποιηθεί, όπως ορίζεται από την εκφώνηση. Η συνάρτηση διαθέτει δικλείδα ασφαλείας σε περίπτωση που το όρισμα *mxid* δεν ανήκει σε κάποια από τις κλάσεις μεταβλητών που περιγράφονται στην εκφώνηση, δηλαδή:

- Ακέραιος αριθμός
- Αλφαριθμητικό (*String*)
- Αραιό μητρώο

Η συνάρτηση έπειτα υπολογίζει τις ζητούμενες μετρικές:

- $rnnz = \frac{nnz(A^{(k)})}{nnz(A)}$
- $rerr = \frac{||A - A^{(k)}||}{||A||}$, ως προς τη νόρμα *Frobenius*

$rnnz$ Η μετρική $rnnz$ αυξάνεται καθώς το k αυξάνεται. Ουσιαστικά είναι ανάλογη του k. Αυτό συμβαίνει καθώς όσο αυξάνεται το εύρος ζώνης (*bandwidth*) του μητρώου, δηλαδή όσο αυξάνεται το k τόσο θα αυξάνεται και ο αριθμός των μη μηδενικών στοιχείων του $A^{(k)}$, τείνοντας στον αριθμό μη μηδενικών στοιχείων του A.

$rerr$ Η μετρική $rerr$ μειώνεται καθώς το k αυξάνεται κι άρα η σχέση μεταξύ $rerr$ και k χαρακτηρίζεται ως αντιστρόφως ανάλογη. Αυτό συμβαίνει καθώς όσο αυξάνεται το k και συνεπώς "συμπληρώνεται" το $A^{(k)}$, τόσο ελαττώνεται και το σχετικό σφάλμα μεταξύ $A^{(k)}$ και A.

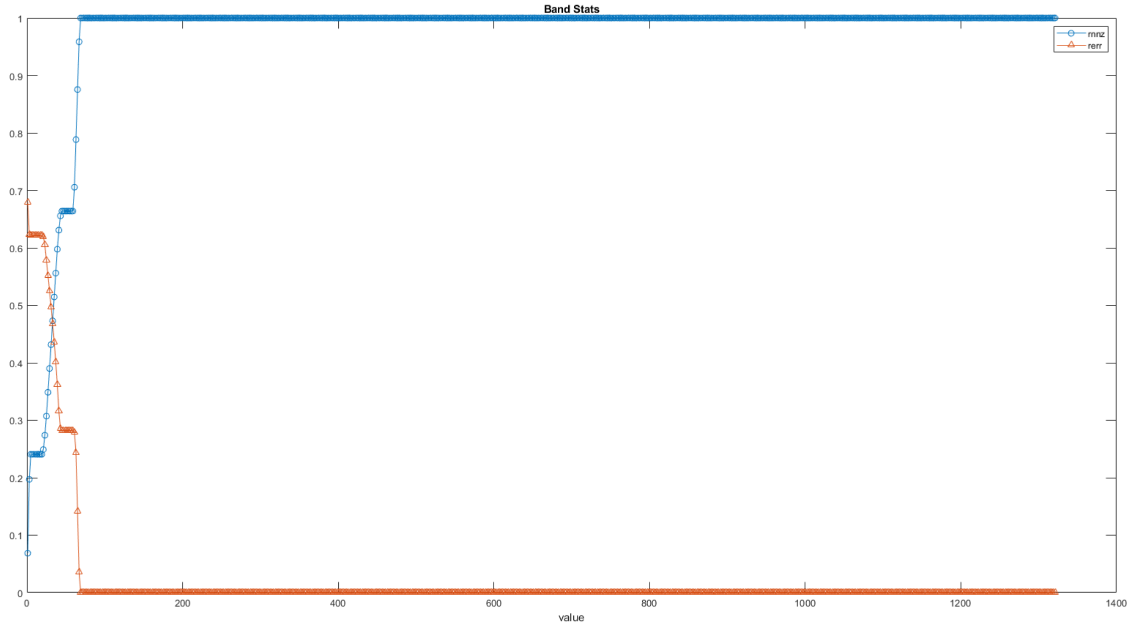
Σε κάποιο σημείο, πριν από το μέγιστο k, οι μετρικές $rnnz$ και $rerr$ σταθεροποιούνται. Αυτό είναι το σημείο όπου δεν υπάρχουν άλλα μη μηδενικά στοιχεία στο δεδομένο μητρώο εκτός του εκάστοτε εύρους ζώνης. Οι μετρικές $rnnz$ και $rerr$ κυμαίνονται στο διάστημα $[0, 1]$, όπου:

- 0 σημαίνει ότι όλα τα μη μηδενικά στοιχεία του μητρώου A βρίσκονται εκτός του εκάστοτε εύρους ζώνης
- 1 σημαίνει ότι όλα τα μη μηδενικά στοιχεία του μητρώου A βρίσκονται εντός του εκάστοτε εύρους ζώνης

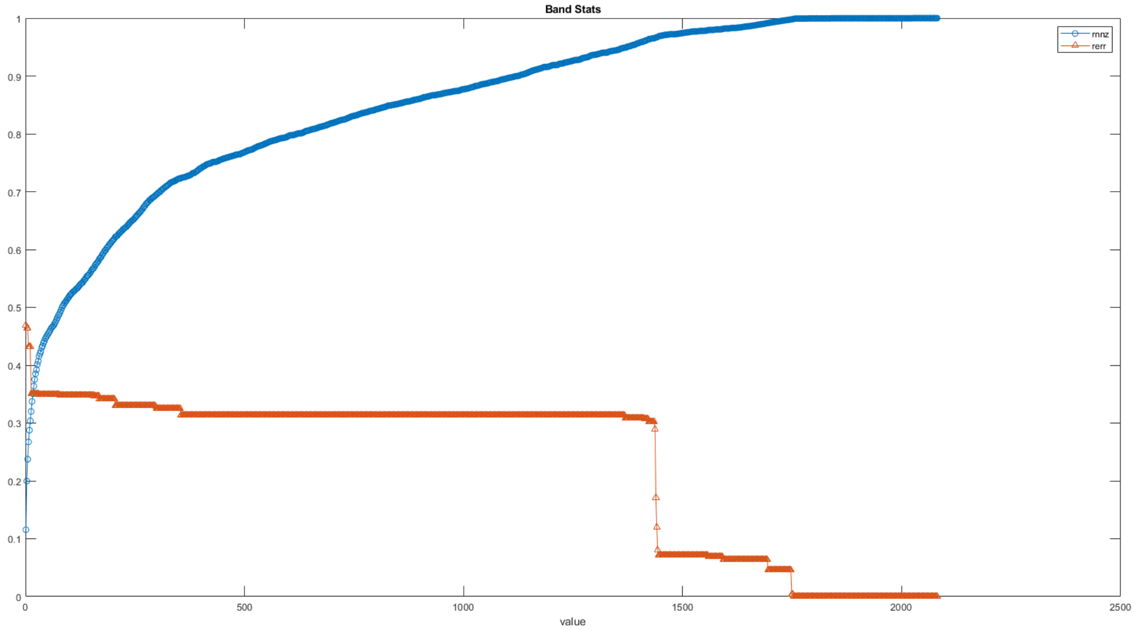
Σχολιασμός δοκιμαστικών μητρώων Τα μητρώα στα οποία δοκιμάστηκε η συνάρτηση `band_stats` ήταν:

- Το μητρώο που επέστρεψε η εντολή `gallery('wathen',10,20)`
- Το μητρώο που επέστρεψε η εντολή `ssget('Rajat/rajat04')`
- Το μητρώο που επέστρεψε η εντολή `ssget(1 + mod(4336, 2892))`[1]

Τα αποτελέσματα της συνάρτησης `band_stats` με όρισμα τα παραπάνω μητρώα δίνονται στα σχήματα 4.1, 4.2 και 4.3.

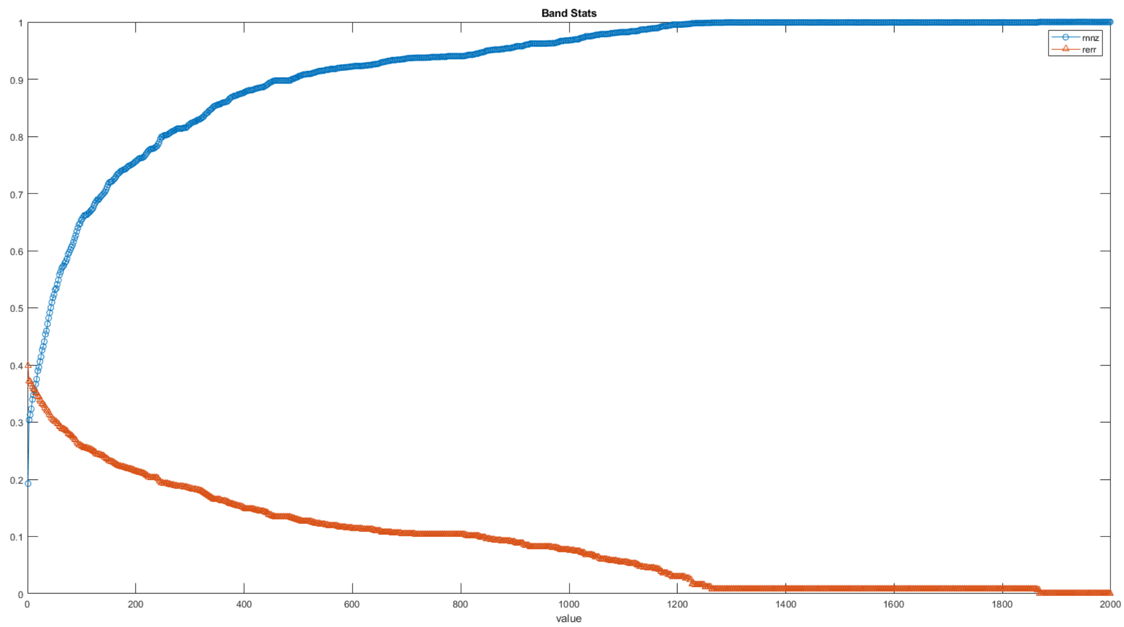


Σχήμα 4.1: Οπτικοποίηση των *rnnz* και *rerr* για το μητρώο που επέστρεψε η εντολή `gallery('wathen',10,20)`



Σχήμα 4.2: Οπτικοποίηση των *rnnz* και *rerr* για το μητρώο που επέστρεψε η εντολή `ssget('Rajat/rajat04')`

[1] Ο Αριθμός Μητρώου του φοιτητή είναι 1054336, κι άρα τα 4 λιγότερο σημαντικά ψηφία είναι 4336.

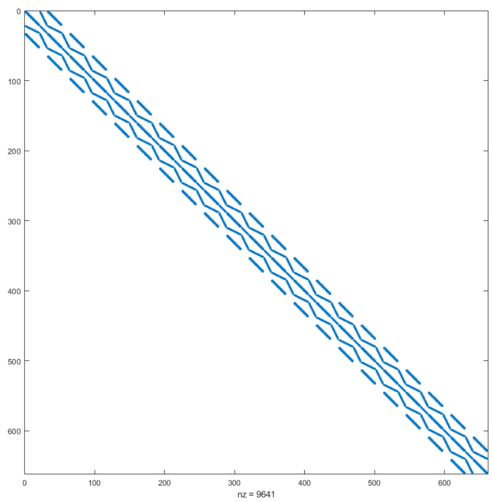


Σχήμα 4.3: Οπτικοποίηση των $rnnz$ και $rerr$ για το μητρώο που επέστρεψε η εντολή `ssget(1 + mod(4336, 2892))`

Αυτό που απορρέει από τα σχήματα 4.1, 4.2 και 4.3 είναι:

εύρος ζώνης μητρώου `gallery('wathen',10,20)` $<$
εύρος ζώνης μητρώου `ssget(1 + mod(4336, 2892))` $<$
εύρος ζώνης μητρώου `ssget('Rajat/rajat04')`

Μάλιστα στο μητρώο που προκύπτει από την εντολή `gallery('wathen',10,20)` είναι μητρώο ζώνης, όπως φαίνεται και στο σχήμα 4.4.



Σχήμα 4.4: Οπτικοποίηση των μη μηδενικών στοιχείων του μητρώου `gallery('wathen',10,20)`

Ο εξεταστής μπορεί να εκτελέσει το *script* `test_band_stats` το οποίο καλεί τη συνάρτηση `band_stats` με ορίσματα τα μητρώα που περιγράφηκαν παραπάνω. Αν και τα μητρώα είναι τετραγωνικά, γράφηκε η συνάρτηση `genmat2sqmat`, η οποία μετατρέπει ένα μητρώο τυχαίων διαστάσεων σε τετραγωνικό μητρώο. Η συνάρτηση `genmat2sqmat` φαίνεται στον κώδικα 4.1.

Listing 4.1: Η συνάρτηση `genmat2sqmat`

```
function [P] = genmat2sqmat(P)
% Author : A. KARATZAS , AM 1054336 , Date : 15/02/2021
%
% GENMAT2SQMAT Converts a general matrix to a square matrix by cutting off
%              rows or columns.
%
%   Usage GENMAT2SQMAT(P) where:
%         P - The given (unfiltered) matrix
%
%   Returns [P] where:
%         P - The filtered matrix

    %% Fine tune matrix
    [rows, cols] = size(P);
    % make matrix a square matrix
    if rows > cols
        P = P(1:cols, 1:cols);
    elseif rows < cols
        P = P(1:rows, 1:rows);
    end
end
```

5. Επαναληπτικές μέθοδοι

Οι επαναληπτικές μέθοδοι σχεδιάστηκαν για την επίλυση γραμμικού συστήματος $A \cdot x = b$. Το πλεονέκτημα αρχίζει και φαίνεται ωστόσο σε πολύ μεγάλα μητρώα. Σε πολύ μεγάλα μητρώα, οι παραδοσιακές μέθοδοι επίλυσης συστήματος, όπως η Gram–Schmidt, είναι εξαιρετικά απαιτητικές ως προς τη μνήμη. Οι επαναληπτικές μέθοδοι είναι *matrix-free* το οποίο σημαίνει ότι δεν απαιτούν τα στοιχεία του μητρώου παρά μόνον τη δυνατότητα του υπολογισμού του γινομένου του μητρώου με διάνυσμα. Αυτό κάνει αυτές τις μεθόδους και σχετικά κοστοβόρες.

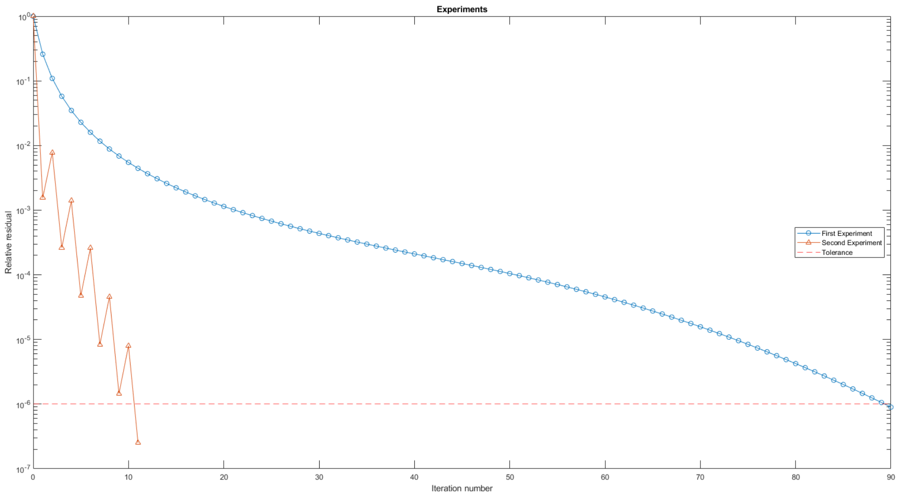
Υποερώτημα 5.1

Ειδικά μητρώα

Σε αυτό το ερώτημα, δημιουργήθηκαν 2 διαγώνια μητρώα με ειδικά χαρακτηριστικά:

- Το μητρώο που χρησιμοποιήθηκε στο 1^{o} πείραμα είναι διαγώνιο μητρώο και τα στοιχεία του βρίσκονται στο διάστημα $[1, 500]$ διατεταγμένα σε αύξουσα σειρά.
- Το μητρώο που χρησιμοποιήθηκε στο 2^{o} πείραμα είναι διαγώνιο μητρώο αλλά δεν εμφανίζει μια καθολική συσχέτιση των στοιχείων του. Αντιθέτως τα στοιχεία $A(1:250, 1:250)$ βρίσκονται στο διάστημα $[1, 2]$, απέχουν περίπου 0.040 το ένα από το άλλο, ενώ τα στοιχεία $A(251:500, 251:500)$ βρίσκονται στο διάστημα $[1000, 1001]$ κι απέχουν ξανά περίπου 0.040 το ένα από το άλλο. Τα στοιχεία είναι διατεταγμένα σε αύξουσα σειρά στη διαγώνιο του μητρώου.

Για το ερώτημα αυτό δημιουργήθηκε το *script* `special_matrices`. Το ζητούμενο διάγραμμα με την πορεία της σύγκλισης ως προς αριθμό των επαναλήψεων σε σχέση με τη νόρμα-2 του σχετικού υπολοίπου φαίνεται στο σχήμα 5.1.



Σχήμα 5.1: Διάγραμμα σύγκλισης ως προς αριθμό των επαναλήψεων σε σχέση με τη νόρμα-2 του σχετικού υπολοίπου. Η νόρμα-2 του σφάλματος για το 1^{o} μητρώο $= 8.978999536359822e-07$ και η νόρμα-2 του σφάλματος για το 2^{o} μητρώο $= 2.520355886994505e-07$.

Η πρώτη παρατήρηση είναι ότι και στα 2 πειράματα, η `pcg` συγκλίνει. Το πρώτο περίεργο φαινόμενο είναι ότι στο δεύτερο πείραμα, η πορεία σύγκλισης της `pcg` δεν είναι γνησίως φθίνουσα, αλλά εμφανίζει *peaks* ανά 2 επαναλήψεις. Το δεύτερο περίεργο φαινόμενο είναι πως ενώ οι ιδιοτιμές για το μητρώο του πρώτου πειράματος κυμαίνονται σε μικρότερο διάστημα σε σχέση με το μητρώο του δεύτερου πειράματος, το τελευταίο συγκλίνει 9 φορές πιο γρήγορα. Συγκεκριμένα, παρόλο που $\frac{\max eig(A_1)}{\min eig(A_1)} < \frac{\max eig(A_2)}{\min eig(A_2)}$, η `pcg` με το μητρώο του $2^{ου}$ πειράματος συγκλίνει 9 φορές πιο γρήγορα από τη `pcg` με το μητρώο του $1^{ου}$ πειράματος. Η εξήγηση πίσω από αυτά τα "περίεργα" φαινόμενα είναι περίπλοκη. Ωστόσο, ιδιαίτερα για το δεύτερο φαινόμενο θα μπορούσε να ειπωθεί πως επειδή οι ιδιοτιμές του μητρώου στο δεύτερο πείραμα είναι κατά (2) συστάδες αρκετά κοντά μεταξύ τους, με τη μια συστάδα να έχει ιδιοτιμές στο διάστημα $[1, 2]$ και τη δεύτερη συστάδα στο διάστημα $[1000, 1001]$, η `pcg` το "εκμεταλλεύεται" και συγκλίνει σχετικά γρήγορα.

Τα αποτελέσματα είναι αρκετά ακριβή και στις 2 περιπτώσεις με το δεύτερο πείραμα να προσεγγίζει ελάχιστα καλύτερα τη λύση. Όπως περιγράφηκε και παραπάνω η ταχύτητα σύγκλισης στο $2^{ο}$ πείραμα είναι καλύτερη από την ταχύτητα σύγκλισης στο $1^{ο}$ πείραμα, καθώς ο αριθμός των επαναλήψεων για το $1^{ο}$ πείραμα είναι 90 ενώ ο αριθμός των επαναλήψεων για το $2^{ο}$ πείραμα είναι 11. Τέλος, ο αριθμός των MV στη `pcg` χωρίς *preconditioner* και με x_0 το μηδενικό διάνυσμα είναι ίσος με $1 + 1 \cdot iterations$. Επομένως:

- Στο $1^{ο}$ πείραμα: Αριθμός πράξεων $MV = 1 + 1 \cdot 90 = 91$.
- Στο $2^{ο}$ πείραμα: Αριθμός πράξεων $MV = 1 + 1 \cdot 11 = 12$.

Η πολυπλοκότητα αυτή προκύπτει από τη διατύπωση *Hestenes-Stiefel* [3] για τη μέθοδο *Συζυγών Κλίσεων*, όπως περιγράφεται και στο Κεφάλαιο 11, παράγραφο 3.7 του πανεπιστημιακού συγγράμματος [2]. Η πολυπλοκότητα αυτή επιβεβαιώνεται και μελετώντας τον κώδικα που εκτελείται καλώντας τη `pcg` στη **MATLAB**. Για παράδειγμα, η **MATLAB** εκτελεί τον κώδικα 5.1 όταν καλείται η `pcg` για το $1^{ο}$ μητρώο[1].

Listing 5.1: Ο κώδικας `pcg` για το $1^{ο}$ πείραμα

```

%% Test first experiment
n = 500;
A = spdiags([1:n]', [0], n, n);
xsol = ones(n, 1);
b = A * xsol;
tol = 1e-6;
maxit = 4 * n;

%% Compute PCG without preconditioner
n2b = norm(b);                      % Norm of rhs vector, b
x = zeros(n,1);

% Set up for the method
flag = 1;
r = b - A * x;
normr = norm(r);                    % Norm of residual
normr_act = normr;
resvec = zeros(maxit+1,1);          % Preallocate vector for norm of residuals
resvec(1,:) = normr;                % resvec(1) = norm(b-A*x0)
normrmin = normr;                   % Norm of minimum residual
rho = 1;
stag = 0;                           % stagnation of the method
moresteps = 0;
maxstagsteps = 3;

% loop over maxit iterations (unless convergence or failure)
for ii = 1 : maxit
    y = r;
    z = y;
    rho1 = rho;
    rho = r' * z;
    if (ii == 1)
        p = z;
```

[1] Ο κώδικας 5.1 ανακτήθηκε από τον πηγαίο κώδικα της συνάρτησης `pcg` της **MATLAB** διαγράφοντας όλα τα ανενεργά σημεία.

```
    else
        beta = rho / rho1;
        p = z + beta * p;
    end

    q = A * p;
    pq = p' * q;
    alpha = rho / pq;

    % Check for stagnation of the method
    if (norm(p)*abs(alpha) < eps*norm(x))
        stag = stag + 1;
    else
        stag = 0;
    end

    x = x + alpha * p;              % form new iterate
    r = r - alpha * q;
    normr = norm(r);
    normr_act = normr;
    resvec(ii+1,1) = normr;

    % check for convergence
    if (normr ≤ tolb || stag ≥ maxstagsteps || moresteps)
        r = b - A * x;
        normr_act = norm(r);
        resvec(ii+1,1) = normr_act;
        if (normr_act ≤ tolb)
            flag = 0;
            iter = ii;
            break
        else
            if stag ≥ maxstagsteps && moresteps == 0
                stag = 0;
            end
            moresteps = moresteps + 1;
        end
    end
    if (normr_act < normrmin)      % update minimal norm quantities
        normrmin = normr_act;
    end
end                                % for ii = 1 : maxit

relres = normr_act / n2b;
resvec = resvec(1:ii+1,:);

%% Plot
figure;
semilogy(0:length(resvec)-1, resvec/norm(b),'-o')
yline(tol,'r--');
legend('First Experiment','Tolerance','Location','East')
xlabel('Iteration number')
ylabel('Relative residual')
title('Experiments')
```

Υποερώτημα 5.2

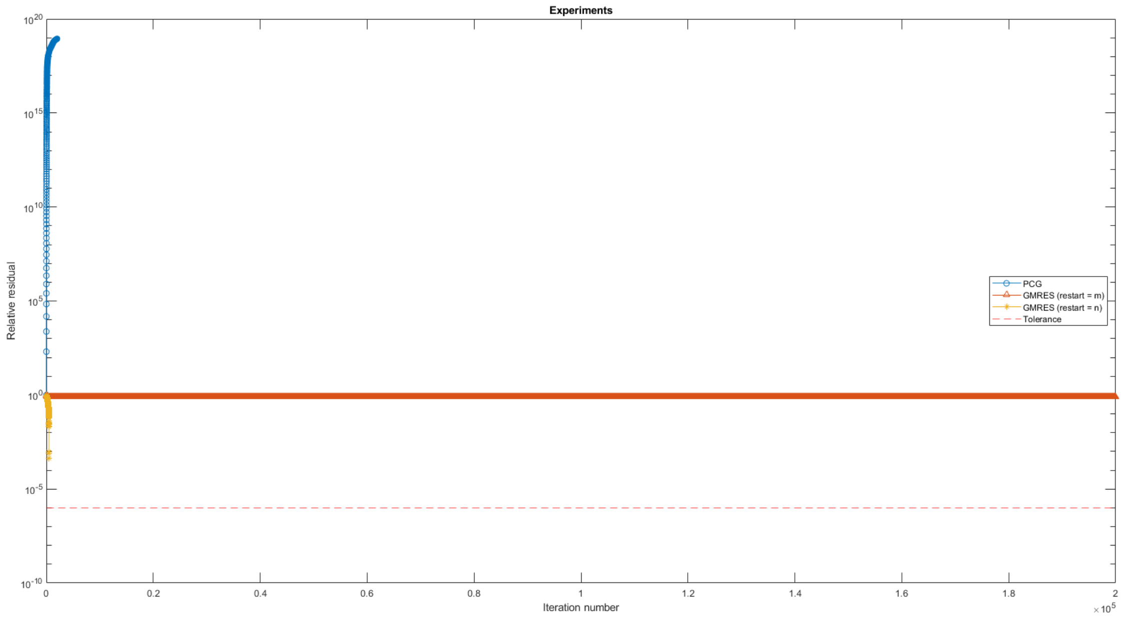
Τυχαία μητρώα

Για το ερώτημα αυτό δημιουργήθηκε το *script* `random_matrices`. Ουσιαστικά, σε αυτό το ερώτημα γίνεται σύγκριση μεταξύ μερικών επαναληπτικών μεθόδων που διδάχθηκαν στο μάθημα και της

LU[2]. Οι επαναληπτικές μέθοδοι που εξετάζονται στο παρόν ερώτημα είναι:

- Η `pcg`
- Η `gmres`

Στη `gmres` δοκιμάζεται και η τεχνική της επανεκκίνησης. Τα αποτελέσματα για ένα τυχαίο γραμμικό σύστημα, όπως περιγράφεται στην εκφώνηση φαίνονται στο σχήμα 5.2.



Σχήμα 5.2: Διάγραμμα σύγκλισης ως προς αριθμό των επαναλήψεων σε σχέση με τη νόρμα-2 του σχετικού υπολοίπου. Η νόρμα-2 του σφάλματος για το πείραμα με τη `pcg` είναι 1.0. Η νόρμα-2 του σφάλματος για το πείραμα με τη `gmres` όπου *restart* $= m = 100$ είναι 0.799775743145601. Η νόρμα-2 του σφάλματος για το πείραμα με τη `gmres` όπου *restart* $= n = 500$ είναι $1.765947828697028e-15$. Η νόρμα-2 του σφάλματος για το πείραμα χρησιμοποιώντας τον τελεστή $\backslash$ είναι $1.409543920058653e-14$.

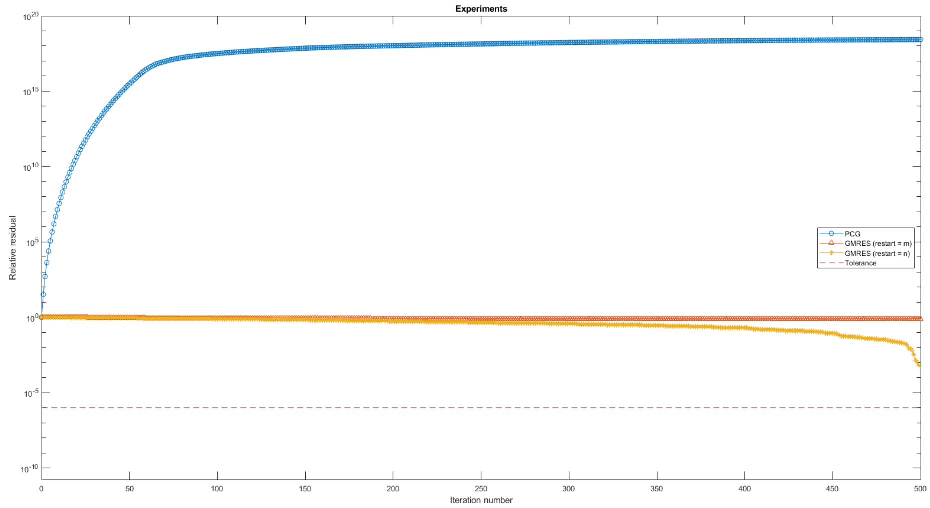
Αξίζει να σημειωθεί πως στη `gmres`, μπορεί να αντικατασταθεί η μεταβλητή `maxit` με τη *default* επιλογή, δηλαδή να εκτελεστούν οι εντολές:

- `[x_gmres_m, flag_gmres_m, ~, iter_gmres_m, resvec_gmres_m] = gmres(A, b, m, tol, [], [], [], []);` και

- `[x_gmres_n, flag_gmres_n, ~, iter_gmres_n, resvec_gmres_n] = gmres(A, b, n, tol, n, [], [], []);` αντίστοιχα,

ανακτώντας τα ίδια αποτελέσματα ως προς την ακρίβεια προσέγγισης της λύσης, σε πολύ καλύτερο χρόνο όμως. Επίσης, αντικαθιστώντας τις εντολές του πειράματος με τις παραπάνω, το ζητούμενο διάγραμμα γίνεται λίγο πιο ευκρινές (βλέπε σχήμα 5.3).

[2] Ο τελεστής $\backslash$ ή αλλιώς `mldivide` χρησιμοποιεί *LU* για την επίλυση γραμμικού συστήματος σε περίπτωση που το μητρώο είναι ένα γενικό (τυχαίο) τετραγωνικό μητρώο.



Σχήμα 5.3: Διάγραμμα σύγκλισης ως προς αριθμό των επαναλήψεων σε σχέση με τη νόρμα-2 του σχετικού υπολοίπου. Η νόρμα-2 του σφάλματος για το πείραμα με τη `pcg` είναι 1.0. Η νόρμα-2 του σφάλματος για το πείραμα με τη `gmres` όπου *restart* $= m = 100$ είναι 0.779670543887418. Η νόρμα-2 του σφάλματος για το πείραμα με τη `gmres` όπου *restart* $= n = 500$ είναι $1.485670181776399e-15$. Η νόρμα-2 του σφάλματος για το πείραμα χρησιμοποιώντας τον τελεστή $\backslash$ είναι $1.233987455576124e-14$.

Αρχικά, είναι εμφανές ότι η `pcg` δεν έχει καθόλου καλή απόδοση. Αυτό συμβαίνει καθώς από θεωρία είναι γνωστό ότι η `pcg` χρησιμοποιείται μόνο σε περιπτώσεις που το μητρώο εισόδου είναι *Συμμετρικό Θετικά Ορισμένο*. Στο πείραμα που εκτελείται για το συγκεκριμένο υποερώτημα, το μητρώο είναι τετραγωνικό χωρίς καμία παραπάνω ιδιότητα. Για αυτό και η `pcg` σταματάει να εκτελείται αρκετά σύντομα, επιστρέφοντας κωδικό 4, που σημαίνει ότι το μητρώο αποκλίνει και οι λύσεις είναι κοντά στα μέγιστα όρια τιμών (λίγο πριν την υπερχείλιση). Η επόμενη παρατήρηση είναι ότι η `gmres` με *restart* ίσο με m έχει χειρότερη απόδοση από τη *gmres* χωρίς *restart*. Αυτό συμβαίνει καθώς η τεχνική της επανεκκίνησης δημιουργήθηκε γιατί μπορεί να μην υπάρχει χώρος για την αποθήκευσης βάσης V_m αρκετά μεγάλης ώστε να παράγει ικανοποιητική προσέγγιση στη λύση. Στο πείραμα αυτό όμως δεν υπάρχει τέτοιο πρόβλημα, κι άρα υπεισέρχονται μόνο τα μειονεκτήματα της τεχνικής αυτής. Επομένως, χάνεται η μονοτονικότητα του σφάλματος ως προς τη νόρμα A. Τέλος, ως προς τη `gmres` χωρίς επανεκκίνηση, οι αποδόσεις είναι καλύτερες από πριν, όπως και ήταν αναμενόμενο από τη θεωρία. Ωστόσο, φαίνεται ότι ενώ προς το τέλος το σχετικό σφάλμα πλησιάζει το επιθυμητό κατώφλι, δεν το ικανοποιεί. Αυτό σημαίνει ότι ίσως χρειάζονται περισσότερες επαναλήψεις. Ένας άλλος τρόπος όμως που θα μπορούσε να μειώσει αρκετά τον αριθμό των επαναλήψεων είναι η τεχνική της προρύθμισης, παραγοντοποιώντας το αρχικό μητρώο (A) χρησιμοποιώντας *Incomplete LU* [1], καθώς το μητρώο είναι ένα τυχαίο τετραγωνικό μητρώο.

Συνοπτικά, η μόνη περίπτωση όπου η μέθοδος συγκλίνει είναι για τη `gmres` χωρίς επανεκκίνηση. Όπως αναφέρεται και στην εκφώνηση ωστόσο, η μέθοδος έχει χειρότερη απόδοση σε σύγκριση με τον τελεστή της **MATLAB** $\backslash$. Χαρακτηριστικά:

- Απόδοση `gmres` χωρίς επανεκκίνηση: 0.19243 sec.
- Απόδοση *mldivide* (τελεστής $\backslash$): 0.00259 sec.

Αξίζει να σημειωθεί πως η ακρίβεια της λύσης που επιστρέφει η `gmres` χωρίς επανεκκίνηση είναι στο ίδιο επίπεδο με αυτήν της *mldivide*:

- Το σχετικό σφάλμα για τη `gmres` χωρίς επανεκκίνηση είναι $1.5716e-15$
- Το σχετικό σφάλμα για τη *mldivide* είναι $1.1445e-14$

Τέλος, η `gmres` με επανεκκίνηση φαίνεται να έχει καλύτερο χρόνο από τη `gmres` χωρίς επανεκκίνηση, όπως και ήταν αναμενόμενο από τη θεωρία[3].

MVs Ως προς τα *MV*:

- Η *GMRES* έχει $MV = 1 + 2 \cdot iterations$
- Η *PCG* έχει $MV = 1 + 1 \cdot iterations$
- Η *MLDIVIDE*, δηλαδή ο τελεστής \, έχει $MV = 0$ (όλες οι πράξεις που εκτελούνται είναι μεταξύ διανυσμάτων ή *scalars*.)

Υποερώτημα 5.3

Επίλυση ΜΔΕ

[3]Η pcg ενώ φαίνεται να κάνει καλύτερους χρόνους από όλες τις μεθόδους, δεν ισχύει καθώς προλαβαίνει να κάνει μόνο μερικές επαναλήψεις μέχρι να σταματήσει λόγω απόκλισης από τη λύση.

6. Βιβλιογραφία

[1] M. Benzi. Preconditioning Techniques for Large Linear Systems: A Survey. *Journal of Computational Physics*, 182(2):418–477, Nov. 2002. ISSN 00219991. doi: 10.1006/jcph.2002.7176. URL https://linkinghub.elsevier.com/retrieve/pii/S0021999102971767.

[2] G. H. Golub and C. F. Van Loan. *Matrix computations / Gene H. Golub, Charles F. Van Loan*. Johns Hopkins University Press Baltimore, 1983. ISBN 0801830109 0801830117.

[3] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–435, 1952.

[4] D. T. MacVeigh. Effect of data representation on cost of sparse matrix operations. *Acta Informatica*, 7(4):361–394, Dec 1977. ISSN 1432-0525. doi: 10.1007/BF00289469. URL https://doi.org/10.1007/BF00289469.

[5] M. Newman. *Networks An Introduction*. Oxford University Press, 2010. doi: 10.1093/acprof:oso/9780199206650.001.0001. URL https://oxford.universitypressscholarship.com/10.1093/acprof:oso/9780199206650.001.0001/acprof-9780199206650.

[6] G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 2009. ISBN 9780980232714. URL https://books.google.gr/books?id=M19gPgAACAAJ.