Εργαστηριακή Άσκηση Επιστημονικού Υπολογισμού

Ονοματεπώνυμο: Κωνσταντοπούλου Ευαγγελία ΑΜ: 1059560
Έτος Εισαγωγής: 2017

24 Φεβρουαρίου 2021

Περιεχόμενα

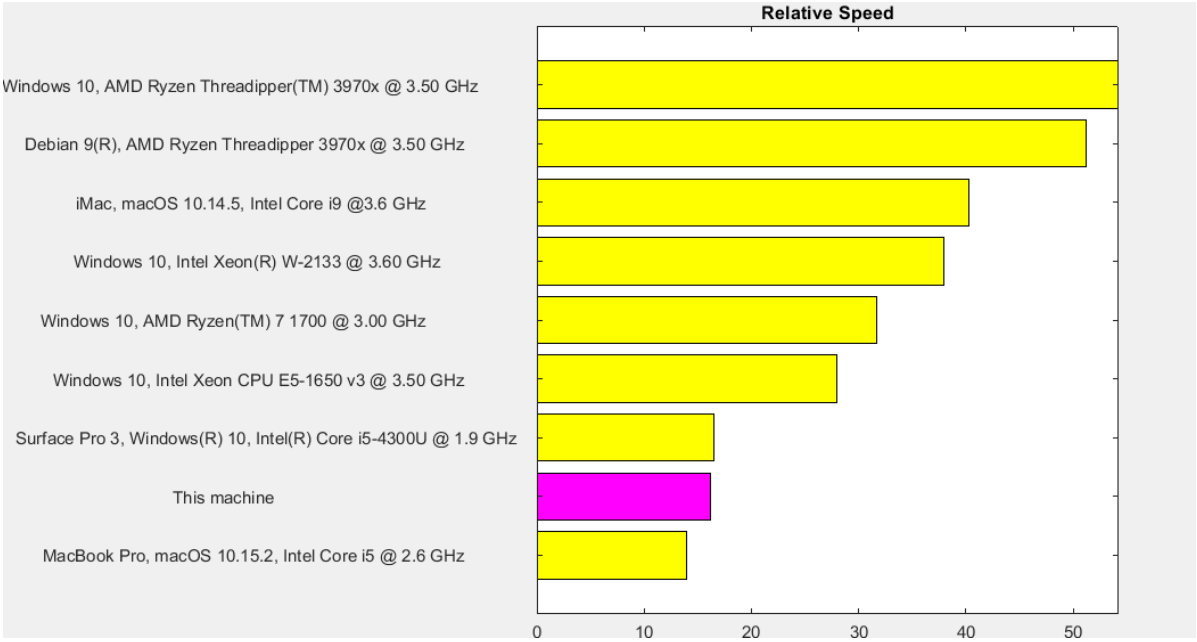
Έναρξη/λήξη εργασίας	13/2/21 - 23/2/21
model	*Laptop Dell Inspiron* 3567[1]
O/S	*Windows* 10 *Home ver.* 10.0.18363 *Build* 18363[2]
processor name	*Intel* $i7-7500U$[3]
processor speed	2.7GHz
number of processors	1
total number of cores	2
total number of threads	4
FMA instruction	FMA 3
L1 cache	2 x 32 KBytes 8-way Instruction, 2 x 32 Kbytes 8-way Data
L2 cache	2 x 256 KBytes 4-way
L3 cache	4 MBytes 16-way
GFlops/s	108[4]
Memory	8 GBs
Memory Bandwidth	34.1 GB/s[5]
MATLAB Version	9.8.0.1323502($R2020a$)[6]
BLAS	Intel(R) Math Kernel Library Version 2019.0.3 Product Build 20190125 for *Intel*(R) 64 *architecture applications*, CNR *branch* $AVX2$[7]
LAPACK	Intel(R) Math Kernel Library Version 2019.0.3 Product Build 20190125 for Intel(R) 64 architecture *applications*, CNR *branch* $AVX2$ *Linear Algebra* $PACKage$ *Version* 3.7.0[8]

0.1 Στοιχεία υπολογιστικού συστήματος

Πηγές:
[1]https://www.msystems.gr/laptops/dell-inspiron-15-3567
[2] Windows System Information
[3] cpuid.com CPUZ
[4] gadgetversus.com/processor/
[5] ark.intel.com
[6] Εντολή version της MATLAB
[7] Εντολή version -blas της MATLAB
[8] Εντολή version -lapack της MATLAB

Computer Type	LU	FFT	ODE	Sparse	2-D	3-D
Windows 10, AMD Ryzen Threadipper(TM) 3970x @ 3.50 GHz	0.2142	0.2931	0.3754	0.4758	0.1914	0.1960
Debian 9(R), AMD Ryzen Threadipper 3970x @ 3.50 GHz	0.2816	0.3059	0.3449	0.4187	0.2827	0.2478
iMac, macOS 10.14.5, Intel Core i9 @3.6 GHz	0.3951	0.4235	0.3112	0.2790	0.7170	0.3641
Windows 10, Intel Xeon(R) W-2133 @ 3.60 GHz	0.4195	0.5219	0.4805	0.5160	0.3000	0.3175
Windows 10, AMD Ryzen(TM) 7 1700 @ 3.00 GHz	0.7770	0.6159	0.5551	0.5734	0.3117	0.3043
Windows 10, Intel Xeon CPU E5-1650 v3 @ 3.50 GHz	0.7741	0.7450	0.5382	0.7103	0.3807	0.3963
Surface Pro 3, Windows(R) 10, Intel(R) Core i5-4300U @ 1.9 GHz	1.7664	1.1723	0.6900	0.6872	0.9974	0.8921
This machine	2.1912	1.1088	0.8098	0.7447	0.7829	0.7537
MacBook Pro, macOS 10.15.2, Intel Core i5 @ 2.6 GHz	1.5141	1.1708	0.5463	0.5959	2.0298	1.4466

Σχήμα 1: Σύγκριση υπολογισμών με την εντολή benchmark

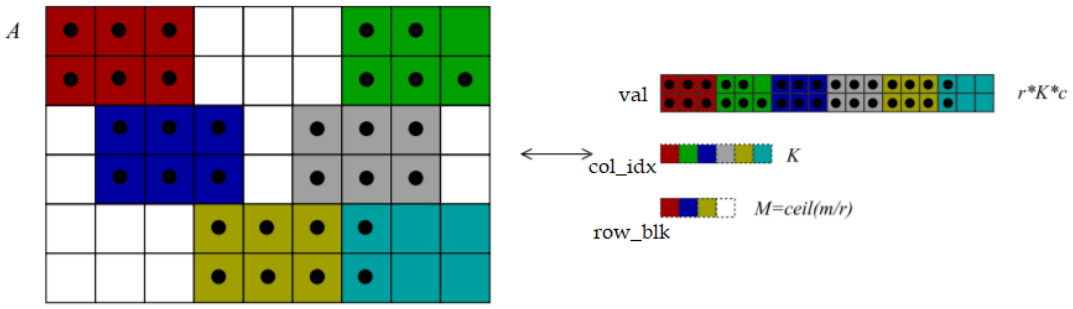


Σχήμα 2: Σύγκριση ταχυτήτων με άλλα συστήματα μέσω της εντολής benchmark

0.2 Αραιή αναπαράσταση BCRS

0.2.1 sp_mx2bcrs

Παρακάτω περιγράφεται η συνάρτηση που παράγει την BCRS αναπαράσταση αραιού μητρώου A, με δεδομένο τη διάσταση των μπλοκ στα οποία θα χωριστεί, nb. Αρχικοποιούμε τις μεταβλητές μας, καθώς και τα διανύσματα εξόδου col_idx, row_blk με μηδενικά. Όσον αφορά το μήκος τους, το row_blk θα έχει μέγεθος όσες οι block γραμμές του μητρώου, δηλαδή τη διάσταση μητρώου A, m, προς nb συν ένα. Στο col_idx δε, θα δώσω το μέγιστο μήκος που μπορεί να πάρει, δηλαδή εάν όλα τα υπομητρώα του A είναι μη-μηδενικά.



Σχήμα 3: Σχηματική αναπαράσταση μεθόδου αποθήκευσης μητρώου, BCRS

Για την εύρεση των μη-μηδενικών μπλοκ και την καταγραφή τους, χρησιμοποιώ δύο for loops, που θα διατρέχουν το μητρώο A με βήμα nb. Καθώς θέλω να διατρέξω το μητρώο κατά γραμμές, η for loop που κοιτάζει τις στήλες θα είναι εσωτερική αυτής για της γραμμές. Ελέγχω το μπλοκ υπομητρώο i,j και αποθηκεύω το πλήθος των μη μηδενικών του στοιχείων στη μεταβλητή nz. Εάν ισχύει ότι nz>0, τότε πρέπει να αποθηκεύσω το υπομητρώο καθώς και τη θέση του στα ζητούμενα εξόδου val, col_idx.

Καθώς το val θα περιέχει υπομητρώα, θα έχει τη μορφή ενός τανυστή με το μέγεθος της τρίτης διάστασης να είναι όσο και το πλήθος των μη μηδενικών υπομητρώων, k. Την μεταβλητή k την αυξάνω κατά ένα κάθε φορά που θα θα πληρούται η συνθήκη if της γραμμής 15.

Στο διάνυσμα col_idx αποθηκεύω σε ποια block στήλη βρίσκεται το αντίστοιχο υπομητρώο του val. Η στήλη μπλοκ αυτή θα ισούται με τον πηλίκο της στήλης του μητρώου A που εξετάζω προς τις διαστάσεις των υπομητρώων nb, στρογγυλεμένο στην άνω ακέραια τιμή. Παρατηρώ πως στο εσωτερικό της if, αυξάνω και μια μεταβλητή με αρχική τιμή 0, nnzbl. Η διαφορά μεταξύ της k και της nnzbl είναι ότι

η k αποθηκεύει το συνολικό πλήθος των μη μηδενικών υπομητρώων, ενώ η nnzbl το πλήθος των μη μηδενικών υπομητρώων σε μία γραμμή-μπλοκ.

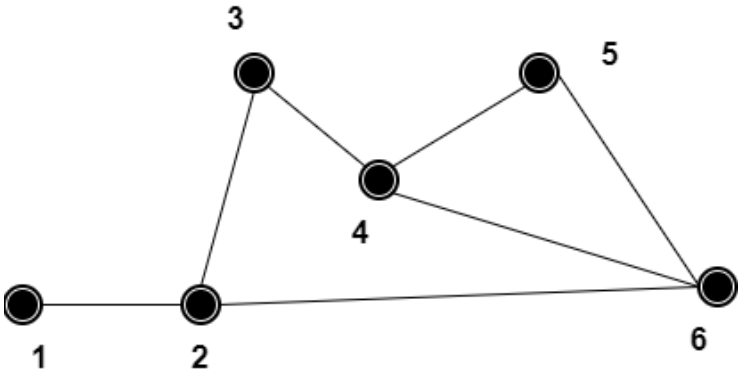
Όταν τελειώσω με τη διάτρεξη μιας μπλοκ-γραμμής θα πρέπει να αλλάξω το διάνυσμα εξόδου row_blk. Για τον υπολογισμό των τιμών αυτού του διανύσματος, χρησιμοποιώ έναν επιπλέον μετρητή ctr. Προφανώς, το πρώτο στοιχείο του διανύσματος row_blk είναι ένα. Την αρχικοποίηση αυτή την έχω κάνει πριν τις λούπες. Για τα επόμενα στοιχεία ισχύει πως το περιεχόμενο της θέσης i+1 του row_blk, ισούται με την τιμή της προηγούμενης θέσης, συν το πλήθος των μη μηδενικών μπλοκ στην τωρινή μπλοκ-γραμμή, το οποίο έχω υπολογίσει στη μεταβλητή nnzbl. Τη διαδικασία αυτή θα την επαναλάβω κάθε φορά που τελειώνω με μια γραμμή, όπου προφανώς θα πρέπει να μηδενίσω την μεταβλητή nnzbl και να αυξήσω τον μετρητή ctr κατά ένα.

Εάν φορτώσετε το μητρώο A.mat που είναι ίδιο με αυτό που δόθηκε στο φροντιστήριο για την αντίστοιχη συνάρτηση μπορείτε να διαπιστώσετε την ορθότητα του κώδικα.

0.3 Τανυστές και διαδρομές

0.3.1 Ερώτημα 1

Έστω ότι έχω το γράφημα της εικόνας, στο οποίο αντιστοιχεί το μητρώο γειτνίασης A.



Σχήμα 4: Γράφημα

A =

0	1	0	0	0	0
1	0	1	0	0	1
0	1	0	1	0	0
0	0	1	0	1	1
0	0	0	1	0	1
0	1	0	1	1	0

Σχήμα 5: Μητρώο γειτνίασης A

Από το μάθημα της γραμμικής άλγεβρας, γνωρίζω ότι:

$\left[A^2\right]_{i,j} = \sum_{k=1}^{n} a_{i,k}a_{k,j} =$ # διαδρομών μήκους 2 (1 στάσης) από το i στο j

Απόδειξη: Για δεδομένα (i,j), κάθε όρος $a_{i,k}a_{k,j}$ είναι 0 ή 1. Η μόνη περίπτωση να είναι 1 είναι όταν $a_{i,k} = 1$ και $a_{k,j} = 1$. Αυτό σημαίνει ότι αμφότερες οι ακμές (i → k) και (k → j). υπάρχουν, επομένως υπάρχει η διαδρομή i → k → j. και ο όρος συνεισφέρει στο άθροισμα. Επομένως η τιμή του αθροίσματος μετρά το συνολικό αριθμό διαδρομών μήκους 2 από το i στο j.

Γενίκευση: Η τιμή $\left[A^2\right]_{i,j}$ είναι ίση με το πλήθος των διαδρομών από i στο j μήκους k (k-1 στάσεις)

Φτιάχνω λοιπόν τη συνάρτηση create_tensor(A,k) που θα παίρνει ως είσοδο μητρώο γειτνίασης A και επιθυμητό άνω όριο μήκους διαδρομών k και θα επιστρέφει τανυστή διαστάσεων n x n x k, $G(:,:,l) = A^l$ για l=1...k.

Ας ελέγξω τα αποτελέσματα που μου δίνει η κλήση της συνάρτησης G=create_tensor(A,3) όπου A το μητρώο γειτνίασης που έδειξα παραπάνω. Μας επιστρέφει τένσορα διαστάσεων 6 x 6 x 3, όπου $G(:,:,2), G(:,:,3)$ μας δείχνουν τις διαδρομές μήκους 2 και μήκους 3 ανάμεσα σε δύο οποιουσδήποτε κόμβους αντίστοιχα.

```
G(:,:,1) =

     0     1     0     0     0     0
     1     0     1     0     0     1
     0     1     0     1     0     0
     0     0     1     0     1     1
     0     0     0     1     0     1
     0     1     0     1     1     0


G(:,:,2) =

     1     0     1     0     0     1
     0     3     0     2     1     0
     1     0     2     0     1     2
     0     2     0     3     1     1
     0     1     1     1     2     1
     1     0     2     1     1     3


G(:,:,3) =

     0     3     0     2     1     0
     3     0     5     1     2     6
     0     5     0     5     2     1
     2     1     5     2     4     6
     1     2     2     4     2     4
     0     6     1     6     4     2
```

Σχήμα 6: Αποτέλεσμα κλήσης συνάρτησης create_tensor

Ας πάρουμε για παράδειγμα το ζεύγος κόμβων i=2, j=1. Στο G(:,:,1)=A έχουμε 1 στη συγκεκριμένη θέση καθώς υπάρχει ακμή που πηγαίνει από το 2 στο 1. Στο G(:,:,2), στην ίδια θέση έχουμε 0 καθώς δεν υπάρχει διαδρομή μήκους 2 που να ξεκινάει από τον κόμβο 2 και να τελειώνει στον κόμβο 1. Στο G(:,:,3), στην ίδια θέση έχουμε 3 καθώς υπάρχουν 3 διαδρομές μήκους 3 που να ξεκινάνε από τον κόμβο 2 και να τελειώνουν στον κόμβο 1. (2→1→2→1, 2→3→2→1, 2→6→2→1).

0.3.2 Ερώτημα 2

Για την υλοποίηση του δεύτερου ερωτήματος, φτιάχνω τη συνάρτηση path_of_pair, η οποία παίρνει ως ορίσματα τον τένσορα που έφτιαξα στο ερώτημα 1, το μέγιστο μήκος διαδρομών που θέλω να υπολογίσω, k, και το ζεύγος κόμβων για τα οποία

θέλω να υπολογίσω διαδρομές μεταξύ τους. Αρχικά, με τη συνάρτηση του MTT, tenmat παράγω το ξεδίπλωμα τρόπου 3 του τανυστή που έφτιαξα προηγουμένως. Καθώς γνωρίζω τη νέα διάταξη που θα έχει το μητρώο, θα ξέρω τη νέα θέση του πλήθους των διαδρομών μήκους 1,2 και 3 αντίστοιχα ανάμεσα στα i=2, j=1. Όντως αν καλέσω την συνάρτηση με την εντολή: sum1=path_of_pair(G,3,2,1), παίρνω το άθροισμα των αριθμών που βρίσκονταν στις θέσεις (i,j)=2,1 του τανυστή G για k=1,2,3. (sum1=4=1+0+3).

0.3.3 Ερώτημα 3

Για την υλοποίηση του δεύτερου ερωτήματος, φτιάχνω τη συνάρτηση path_of_all, η οποία παίρνει ως ορίσματα τον τένσορα που έφτιαξα στο ερώτημα 1 και το μέγιστο μήκος διαδρομών που θέλω να υπολογίσω, k. Και πάλι, με τη συνάρτηση του MTT, tenmat παράγω το ξεδίπλωμα τρόπου 3 του τανυστή που έφτιαξα προηγουμένως. Το ζητούμενο πλήθος των διαδρομών μήκους έως k δεν είναι παρά το άθροισμα των στηλών του νέου μητρώου, για γραμμές από i=1 έως k. Για τον υπολογισμό αυτό θα χρειαστώ το πλήθος των στηλών του νέου μητρώου, που από τη θεωρία γνωρίζω ότι ισούται με το γινόμενο των δύο πρώτων διαστάσεων του τανυστή G. Τα αθροίσματα για κάθε στήλη θα αποθηκεύονται σε ένα διάνυσμα μήκους N, sum2. Επομένως αν θέλω να μάθω το πλήθος των διαδρομών από κόμβο i=3 σε κόμβο j=4, απλά θα κοιτάξω στη θέση (3,4) του διανύσματος εξόδου sum2.

Στο script3 δίνω το μητρώο γειτνίασης A καθώς και τις εντολές κλήσης των path_of_pair, path_of_all για k=2,3 για έλεγχο ορθότητας.

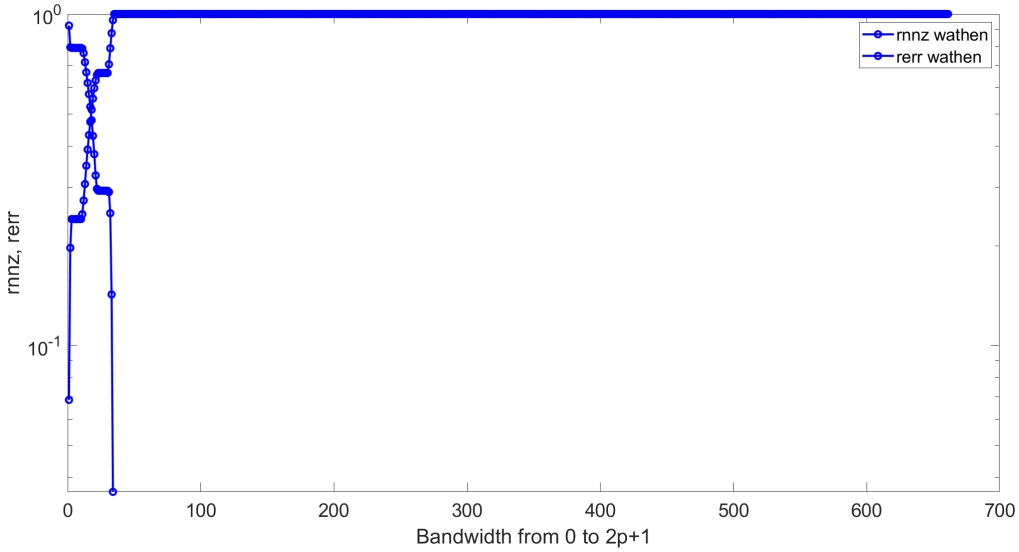
0.4 Στατιστικά μητρώων

Παρακάτω παρατίθεται η υλοποίηση της συνάρτησης P=band_stats(mxpid). Πρώτα θα γίνει έλεγχος για τη φύση του mxid. Εάν το mxid είναι array ή μητρώο, αυτό σημαίνει ότι μία από τις δύο διαστάσεις του θα είναι μεγαλύτερες της μονάδας . Τότε χρησιμοποιείται το ίδιο το mxid. Εάν είναι ακέραιος, τότε με τη βοήθεια της ssget παίρνει το μητρώο από τη συλλογή Suite Sparse με id=mxid. Εάν είναι string, τότε με τη βοήθεια της ssget παίρνει το μητρώο από τη συλλογή Suite Sparse με name=mxid.

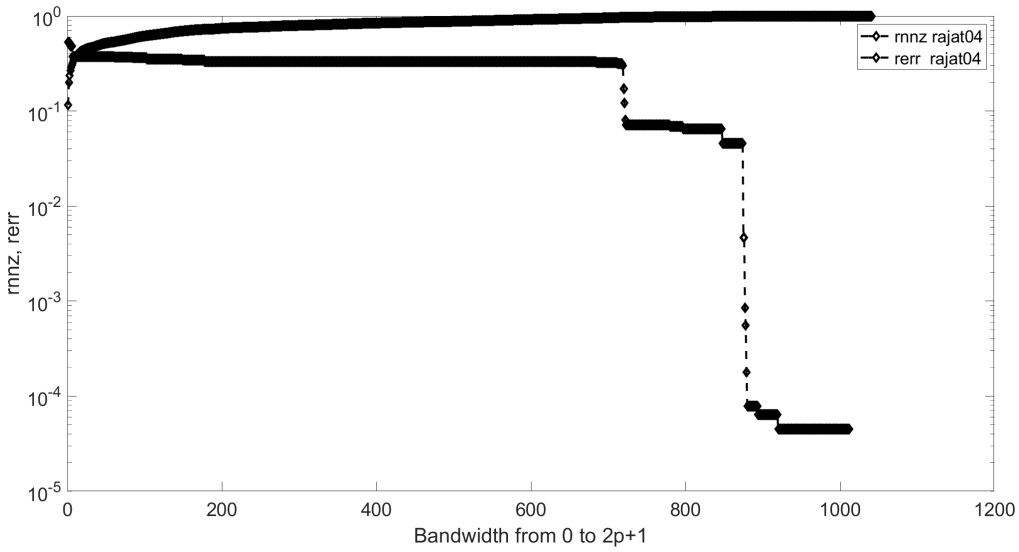
Ορίζω το μητρώο M ίσο με το A. Αυτό το κάνω για να κρατήσω αποθηκευμένη την αρχική μορφή του A ώστε να υπολογίσω τα rnnz, rerr αργότερα. Από τις διευκρινίσεις γνωρίζω ότι το p ισούται με το μέγεθος του μητρώου n. Ο πίνακας P που θα περιέχει τις τιμές rnnz, rerr αρχικοποιείται ως ένα μητρώο με 2 στήλες και γραμμές όσες το μήκος του μητρώου εισόδου.

Θέλω εύρος ζώνης από 1 έως 2p+1 επομένως για τις επαναλήψεις μου θα έχω k από 0 έως p-1. Ανάλογα με την τιμή του p μηδενίζω τις κατάλληλες τιμές εκατέρωθεν των διαγωνίων. Όταν το μητρώο ζώνης μου είναι έτοιμο, θα υπολογίσω τα ζητούμενα rnnz και rerr, σύμφωνα με τους τύπους που δόθηκαν στην εκφώνηση και θα τα αποθηκεύσω στον πίνακα P στις στήλες 1 και 2 αντίστοιχα.

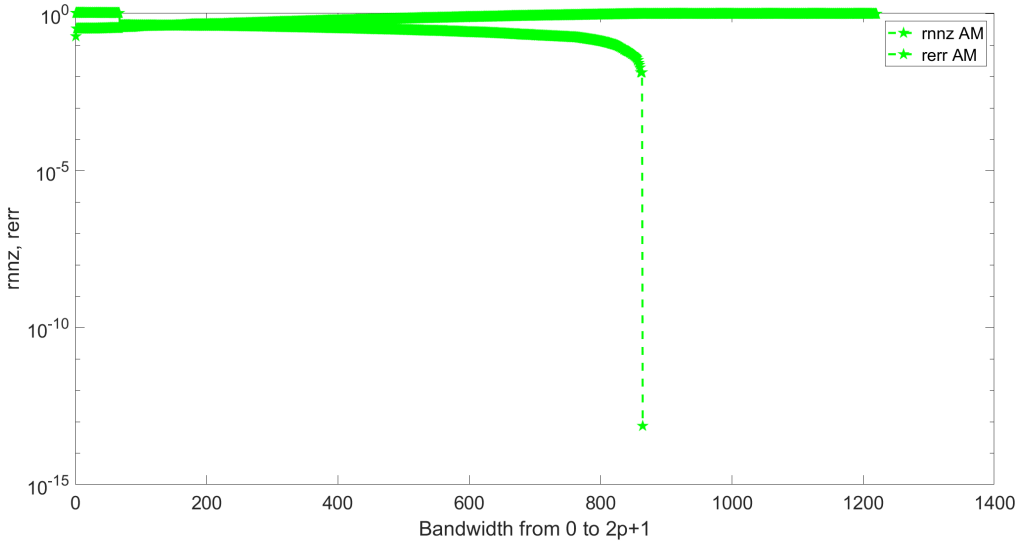
Πάμε να δούμε τα αποτελέσματα που πήρα για τα μητρώα ελέγχου.



Σχήμα 7: Αποτελέσματα για μητρώο wathen



Σχήμα 8: Αποτελέσματα για μητρώο rajat04

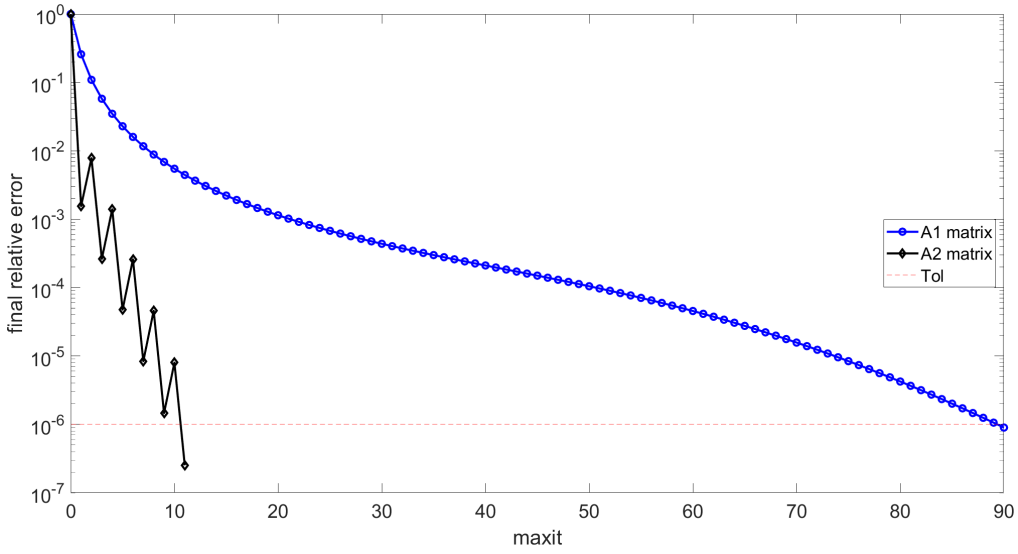


Σχήμα 9: Αποτελέσματα για μητρώο AM

0.5 Επαναληπτικές μέθοδοι

0.5.1 Ειδικά μητρώα

Εάν κοιτάξουμε τις σημαίες fl1, fl2 που μας υποδεικνύουν τη σύγκλιση (ή μη), διαπιστώνουμε ότι και στις δύο περιπτώσεις ο αλγόριθμος συγκλίνει. Από τη θεωρία γνωρίζω πως για την pcg χρειάζομαι μία πράξη MV για κάθε iteration, επομένως για να βρω το πλήθος αυτών θα κοιτάξω τις μεταβλητές it1,it2. Έχω it1=90, it2=11, επομένως για το δεύτερο μητρώο απαιτούνται λιγότερες επαναλήψεις και επομένως λιγότερες πράξεις. Ακόμα και από το διάγραμμα βλέπω πως ο αλγόριθμος για το δεύτερο μητρώο συγκλίνει πιο γρήγορα. Επίσης, από το σχετικό υπόλοιπο βλέπω πως στη δεύτερη περίπτωση έχω μεγαλύτερη ακρίβεια αποτελεσμάτων. Το φαινομενικά περίεργο είναι πως εάν δω τη μορφή των δύο αυτών μητρώων script spied.m) φαίνονται να είναι ίδια, παρόλα αυτά όμως για τον ίδιο αλγόριθμο έχω διαφορετική ταχύτητα σύγκλισης, διαφορετικό πλήθος πράξεων, διαφορετική ακρίβεια υπολογισμού αποτελεσμάτων.



Σχήμα 10: Αριθμός των επαναλήψεων σε σχέση με τη νόρμα-2 του σχετικού υπολοίπου.
Νόρμα-2 του σφάλματος στο τελευταίο βήμα για A1: 8.9790ε-07
Νόρμα-2 του σφάλματος στο τελευταίο βήμα για A2: 2.5204ε-07

Σχήμα 11: Μητρώα A1, A2

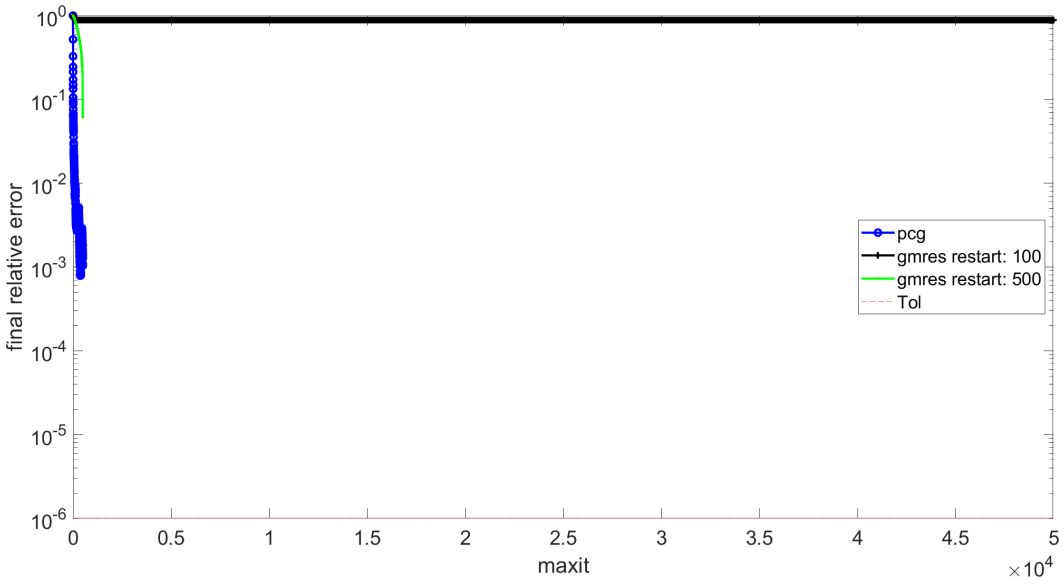
0.5.2 Τυχαία μητρώα

Απ΄ ότι φαίνεται, εάν κοιτάξουμε τις σημαίες fl1, fl2, fl3 που μας υποδεικνύουν τη σύγκλιση (ή μη), μόνο ο τρίτος αλγόριθμος συγκλίνει (fl3=0). Καθώς εδώ οι διαστάσεις του προβλήματος είναι πολύ μεγάλες, η pcg αποτυγχάνει και θα πρέπει να βασιστώ στη gmres. Χρειάζομαι μία πράξη MV για κάθε iteration, επομένως

για να βρω το πλήθος αυτών θα κοιτάξω τη μεταβλητή it3, καθώς μόνο ο τρίτος αλγόριθμος συγκλίνει. Για τη gmres, η μεταβλητή it είναι ένα διάστημα δύο στοιχείων που δείχνει τις [εξωτερικές εσωτερικές] επαναλήψεις αντίστοιχα. Έχω it3=1 500, επομένως για το δεύτερο μητρώο τόσες πράξεις πολλαπλασιασμού ανάμεσα σε μητρώο και διάνυσμα απαιτούνται ως τη σύγκλιση.

Για να αποδείξω πως με την ανάποδη κάθετο παίρνω ακριβή αποτελέσματα με γρηγορότερο τρόπο, μετρώ τον elapsed time των υπολογισμών μέσω tic-toc, και συγκρίνω τη νόρμα 2 του σφάλματος στο τελευταίο βήμα για τις μεθόδους pcg, gmres με το αντίστοιχο σφάλμα της ανάποδης καθέτου. Στον παρακάτω πίνακα παραθέτω τα αποτελέσματα των μετρήσεων μου, όπου $err = \frac{||x-x_{sol}||}{||x_{sol}||}$:

μέθοδος	**ταχύτητα**	**σφάλμα**
pcg	0.177137 sec	0.2870
gmres 100	6.697789 sec	0.7915
gmres 500	0.271201 sec	0
backslash	0.014797 sec	8.5032ε-14

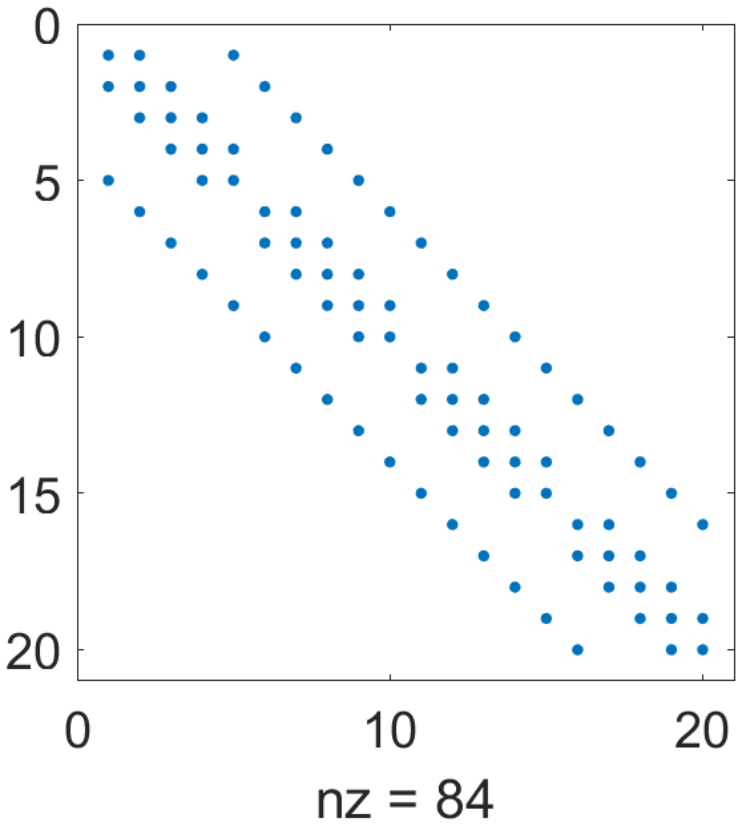


Σχήμα 12: Αριθμός των επαναλήψεων σε σχέση με τη νόρμα-2 του σχετικού υπολοίπου για μεθόδους pcg, gmres restart:100, gmres restart: 500
Νόρμα-2 του σφάλματος στο τελευταίο βήμα για pcg: 0.2870
Νόρμα-2 του σφάλματος στο τελευταίο βήμα για gmres 100: 0.7915
Νόρμα-2 του σφάλματος στο τελευταίο βήμα για gmres 500: 0

0.5.3 Επίλυση ΜΔΕ

Μητρώο συντελεστών Α

Για να φτιάξω το μητρώο συντελεστών Α, χρησιμοποίησα γινόμενα Kronecker, και συγκεκριμένα τα γινόμενα Kronecker ανάμεσα σε ταυτοτικό μητρώο I και το μητρώο B του οποίου τη δομή θα εξηγήσω σε λίγο. Σύμφωνα με την εξίσωση που μας δίνεται, θέλουμε το μητρώο Α να έχει μη μηδενικά στοιχεία στην κύρια διαγώνιο, σε υπερδιαγώνιο και υποδιαγώνιο. Οι υπερ/υπόδιαγώνιοι θα έχουν μόνο τιμές -1, γι΄ αυτό ορίζω το διάνυσμα που θα οριστεί ως αυτές ως r=-ones(). Η κύρια διαγώνιος θα παίρνει τιμές 4 εάν το ω ισούται με 0, ενώ με $4\text{-}w^2 * h^2$ εάν ω=1. Έχοντας υπόψη πως το Α θα προκύψει από την πρόσθεση των δύο γινομένων Kronecker, ορίζω κατάλληλα το διάνυσμα της κύριας διαγωνίου r2.



Σχήμα 13: Η μηδενική δομή του μητρώου συντελεστών Α

myMV

Το μητρώο Α έχει μια συγκεκριμένη δομή, επομένως μπορούμε να αναπαραστήσουμε την πράξη A*z με ένα function handle. Όταν το Α πολλαπλασιάζει ένα διάνυ-

σμα, τα περισσότερα στοιχεία του διανύσματος που προκύπτει είναι μηδενικά, ενώ τα μη μηδενικά αντιστοιχούν στα μη μηδενικά διαγώνια στοιχεία του A. Όπως βλέπουμε και στις εικόνες, το μητρώο A*z μπορεί να γραφτεί ως άθροισμα επιμέρους διανυσμάτων.

Για ω=0:

$$\begin{bmatrix} 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & \dots & 0 \\ & & & & & & & & & & & & & & & \\ & & & & & & & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 4 \end{bmatrix} \begin{bmatrix} z1 \\ z2 \\ z3 \\ z4 \\ z5 \\ z6 \\ . \\ . \\ . \\ z19 \\ z20 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 4z1 & (-1)z2 & (-1)z5 \\ 0 & (-1)z1 & 4z2 & (-1)z3 & (-1)z6 \\ 0 & (-1)z2 & 4z3 & (-1)z4 & (-1)z7 \\ 0 & (-1)z3 & 4z4 & (-1)z5 & (-1)z8 \\ (-1)z1 & (-1)z4 & 4z5 & (-1)z6 & (-1)z9 \\ (-1)z2 & (-1)z5 & 4z6 & (-1)z7 & (-1)z10 \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ (-1)z16 & (-1)z19 & 4z20 & 0 & 0 \end{bmatrix}$$

Σχήμα 14: Αναπαράσταση του μητρώου A σε μορφή αθροίσματος δυανυσμάτων