Επιστημονικός Υπολογισμός

ΕΡΓΑΣΙΑ ΕΞΑΜΗΝΟΥ 2020-2021

Τμήμα Μηχανικών Η/Υ και Πληροφορικής

Ελευθεριάδης Πέτρος

1041741

ΠΕΡΙΕΧΟΜΕΝΑ

1 Εισαγωγικά

1.1 Στοιχεία υπολογιστικού συστήματος

Χαρακτηριστικό	
Έναρξη/λήξη εργασίας	12/01/2021 – 25/01/2021
model	MSI GE60 2OE (Laptop)
O/S	Windows 10 64-bit
processor name	Intel Core i7 4710MQ
Processor speed	2.5 GHz (Base)
total # cores	4
total # theads	8
FMA instruction	yes
L1 cache	L1D: 4x32Kbytes, L1I: 4x32Kbytes | 8way-associat
L2 cache	4x256Kbytes | 8 way associative
L3 cache	6 Mbytes | 12 way associative
Memory	8GB
Memory Bandwidth	25.6 GB/s
MATLAB Version	9.9.0.1467703 (R2020b)
BLAS	Intel(R) Math Kernel Library Version 2019.0.3 Product Build 20190125 for Intel(R) 64 architecture applications, CNR branch AVX2
LAPACK	Intel(R) Math Kernel Library Version 2019.0.3 Product Build 20190125 for Intel(R) 64 architecture applications, CNR branch AVX2, supporting Linear Algebra PACKage Version 3.7.0

Computer Type	LU	FFT	ODE	Sparse	2-D	3-D
Windows 10, AMD Ryzen Threadripper(TM) 3970x @ 3.50 GHz	0.1930	0.1892	0.3545	0.4085	0.1999	0.2188
Debian 9(R), AMD Ryzen Threadripper 3970x @ 3.50 GHz	0.2612	0.1259	0.3393	0.4216	0.3005	0.2809
Windows 10, Intel Xeon(R) W-2133 @ 3.60 GHz	0.4010	0.3255	0.4494	0.5081	0.3484	0.3166
Windows 10, Intel Xeon CPU E5-1650 v3 @ 3.50 GHz	0.4571	0.3189	0.4957	0.4492	0.3445	0.3922
iMac, macOS 10.15.3, Intel Core i9 @3.6 GHz	0.3286	0.2994	0.3307	0.2971	0.8115	0.5337
Windows 10, AMD Ryzen(TM) 7 1700 @ 3.00 GHz	0.7786	0.5169	0.5180	0.5948	0.3184	0.2160
This machine	1.1226	0.6549	0.7576	0.7200	0.6066	0.8906
Surface Pro 3, Windows(R) 10, Intel(R) Core i5-4300U @ 1.9 GHz	1.7749	0.9768	0.7254	0.6882	0.6982	0.6290
MacBook Pro, macOS 10.15.2, Intel Core i5 @ 2.6 GHz	1.5406	0.9646	0.6172	0.6083	2.0698	1.4805

2 Αραιές αναπαραστάσεις πέραν των CSR, CSC - ΥΠΟΧΡΕΩΤΙΚΗ

2.1 sp_mx2bcrs

Η συνάρτηση λειτουργεί ως εξής. Αρχικά κατασκεύασα μια διπλή εμφωλευμενη *for* για τις block γραμμές και block στήλες, οι οποίες αυξάνονται κατά *nb*. Κάθε φορά παίρνουμε το block στο οποίο είμαστε και κοιτάμε αν περιέχει μέσα μη μηδενικά. Αν ναι, τότε το δίνουμε τιμές στο *val* (το ίδιο το block), *col_idx* (τη block στήλη) και *row_blk* (τη block γραμμή) όπως περιγράφεται στη βιβλιογραφία.

Επιπλέον έλαβα υπόψη τις περιπτώσεις μηδενικών γραμμών καθώς και τη περίπτωση οι πρώτες γραμμές να είναι μηδέν.

2.2 spmv_bcrs

Η συνάρτηση λειτουργεί ως εξής. Σε μια μεγάλη for που μετράει τα στοιχεία της *row_blk*, παίρνουμε το πρώτο από αυτά και κοιτάμε πόσα non zero blocks υπάρχουν στη γραμμή αυτή.

Έπειτα για κάθε ένα από αυτά τα blocks στο διάνυσμα *val* κάνουμε πολλαπλασιασμό με τη κατάλληλες *nb* θέσεις του *x* και προσθέτουμε το αποτέλεσμα στη κατάλληλη θέση του v το οποίο αντιπροσωπεύει το A*x.

Παρομοίως έγινε και το A'*x με τη διαφορά ότι τώρα έπαιρνα τιμές από το ανάστροφο του *val* και αντιστράφηκαν οι δείκτες στα διανύσματα *x* και *v*

2.3 ΠΑΡΑΔΕΙΓΜΑΤΑ

Επέλεξα 2 αραιά μητρώα από το sparse.tamu.edu.

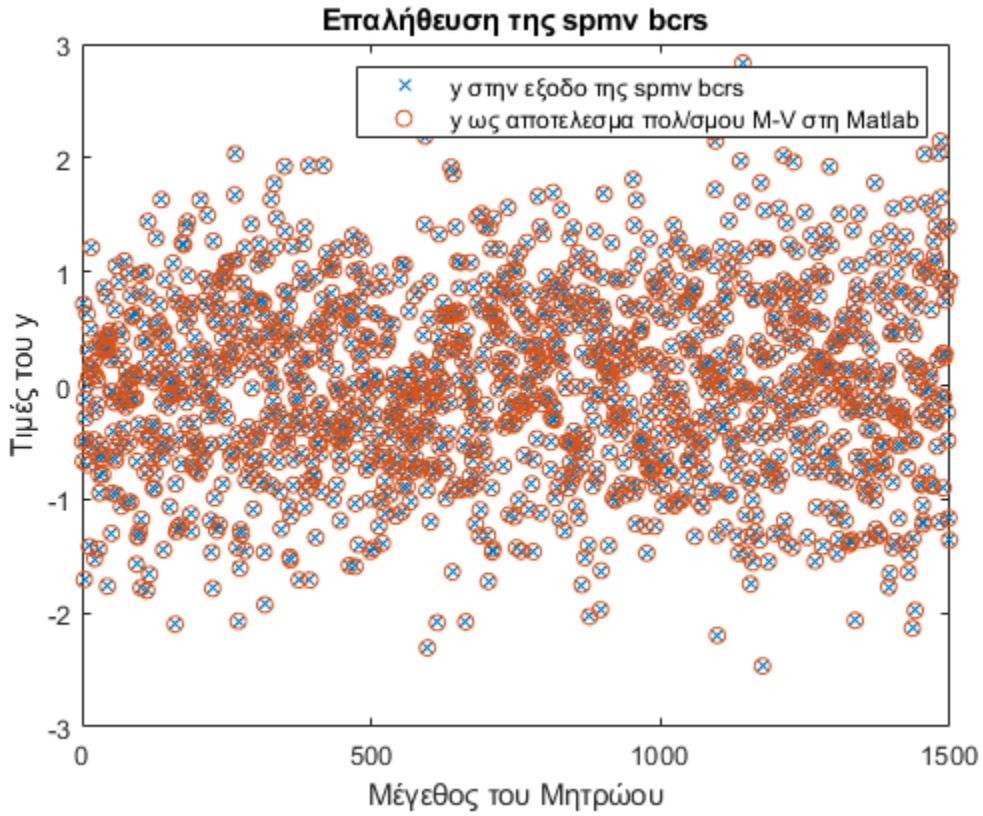
Το πρώτο είναι το olm2000 Bai/olm2000 | SuiteSparse Matrix Collection (tamu.edu) το οποίο είναι ένα αραιό μητρώο 1500x1500 και έχει 7.996 μη μηδενικά στοιχεία

Το δεύτερο είναι το comsol Langemyr/comsol | SuiteSparse Matrix Collection (tamu.edu) με διαστάσεις 2000x2000 και έχει 97.645 μη μηδενικά στοιχεία. A^T

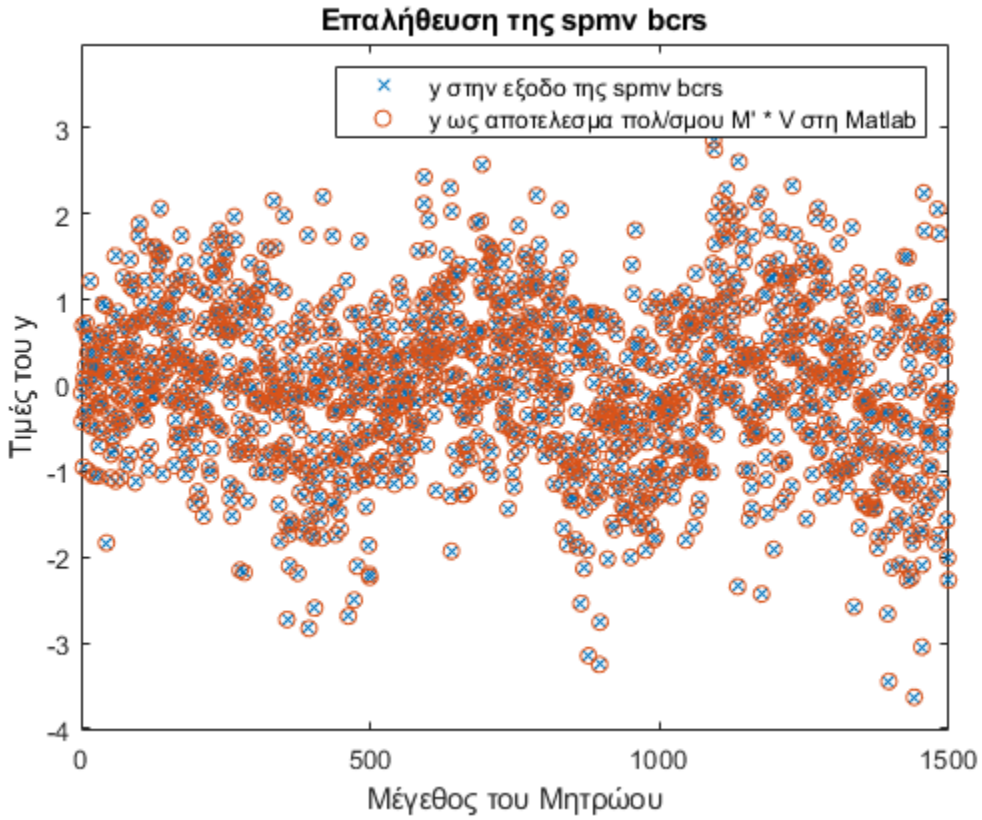
Για το πρώτο μητρώο είχα τα εξής αποτελέσματα. Για τον υπολογισμό του $y = y + A * x$ ο χρόνος που χρειάστηκε ήταν **0.009643 δευτερόλεπτα** (με τη χρήση tic toc) και για τον υπολογισμό του $y = y + A^T * x$ ο χρόνος που χρειάστηκε ήταν **0.0084535 δευτερόλεπτα**.

Για το δεύτερο μητρώο αντίστοιχα οι χρόνοι ήταν **0.311262** και **0.308411 δευτερόλεπτα**.
Ο λόγος που ο χρόνος είναι τόσο αυξημένος σε σχέση με το πρώτο μητρώο είναι πως το δεύτερο μητρώο έχει πολλά περισσότερα μη μηδενικά στοιχεία (+90.000)

Και οι δυο συναρτήσεις λειτουργήσαν σωστά. Ως παράδειγμα, παραθέτω κάτω τα αποτελέσματα του y για το δευτερο μητρωο σε σύγκριση των σωστών αποτελεσμάτων από τη Matlab. Όπως φαίνεται τα αποτελέσματα είναι ακριβώς τα ίδια.



Το y ως αποτέλεσμα της συνάρτησης spmv_bcrs για trans=0 πανω στο γράφημα του y ως αποτέλεσμα υπολογισμών από τη Matlab



Το y ως αποτέλεσμα της συνάρτησης spmv_bcrs για trans=1 πάνω στο γράφημα του y ως αποτέλεσμα υπολογισμών από τη Matlab

3 Τανυστές και Διαδρομές - BONUS

3.1 Κατασκευή Τανυστή

Για τη λειτουργία του κώδικα πρέπει να κάνουμε addpath() τη τοποθεσία του 'TensorToolbox'.

Η κατασκευή του τανυστή είναι απλή. Παίρνουμε όρισμα ένα μητρώο γειτνίασης A και ένα k. Στη συνάρτηση πρώτα αρχικοποιούμε έναν τανυστή με k φλοίδες με μηδενικά για να μπορέσουμε να του περάσουμε τις τιμές του A. Έπειτα σε έναν βρόχο από 1 έως k, δημιουργούμε τις k φλοίδες μια μια από τις δυνάμεις του μητρώου A.

3.2 Εύρεση μιας Διαδρομής

Για τη λύση αυτού του προβλήματος κατασκεύασα μια συνάρτηση που να παίρνει όρισμα τον τανυστή G, τον αριθμό k, το αρχικό σημείο i και τελικό σημείο j. Έπειτα σε έναν βρόχο έως κ, προσθέτω τις τιμές του τανυστή στη ζητούμενη γραμμή και στήλη για όλες τις φλοίδες έως k.

3.3 Εύρεση όλων των Διαδρομών

Στη γενική αυτή περίπτωση πρέπει απλά να προσθέσουμε όλες τις κ φλοίδες του Γ μεταξύ τους. Αυτό γίνεται πολύ εύκολα με τη χρήση της συνάρτησης collapse του Tensor Toolbox. Η συνάρτηση αυτή παίρνει ως όρισμα τη διάσταση των φλοιδών των οποίων θα προσθέσει, οπότε στη δικιά μας περίπτωση δίνουμε το όρισμα 3.

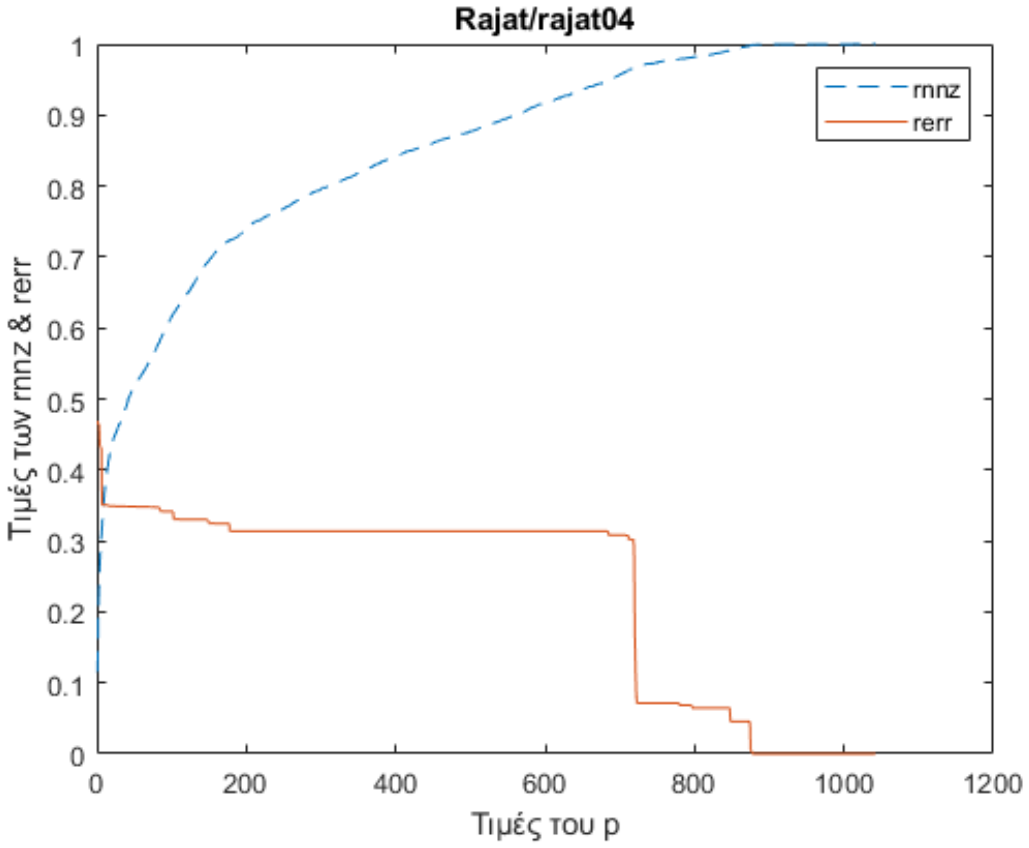
4 Στατιστικά Μητρώων

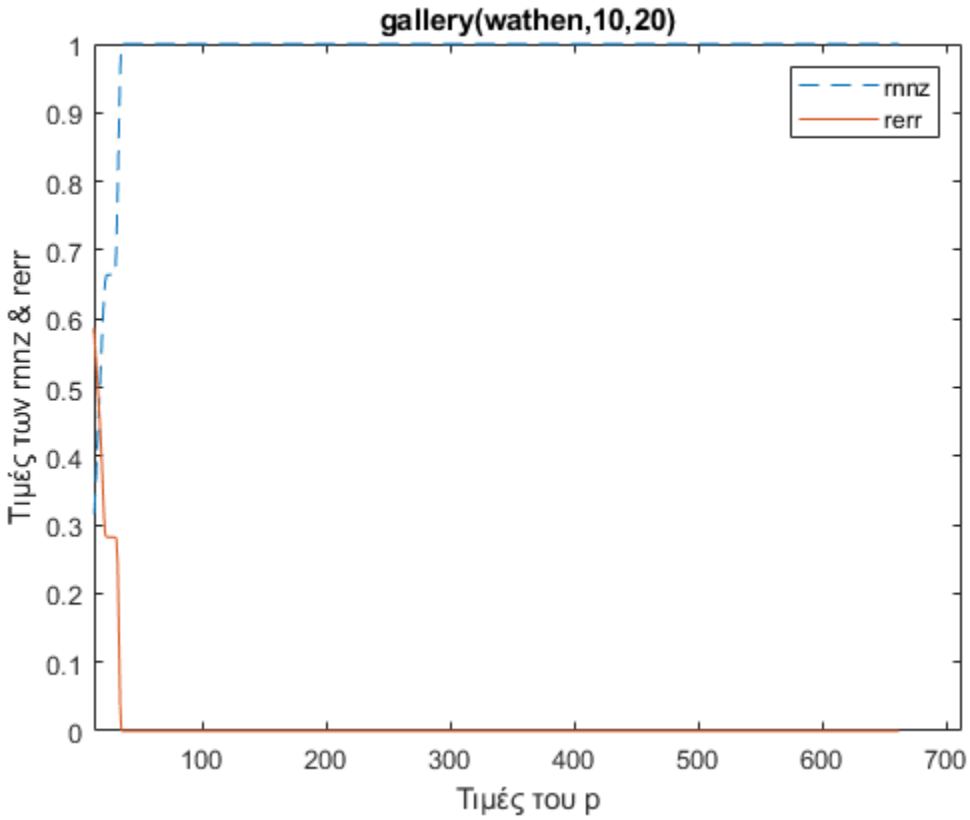
Για τη λειτουργία του κώδικα πρέπει να κάνουμε addpath() τη τοποθεσία του φακέλου 'ssget' που βρίσκεται μέσα στο φάκελο 'SuiteSparse-5.8.1'.

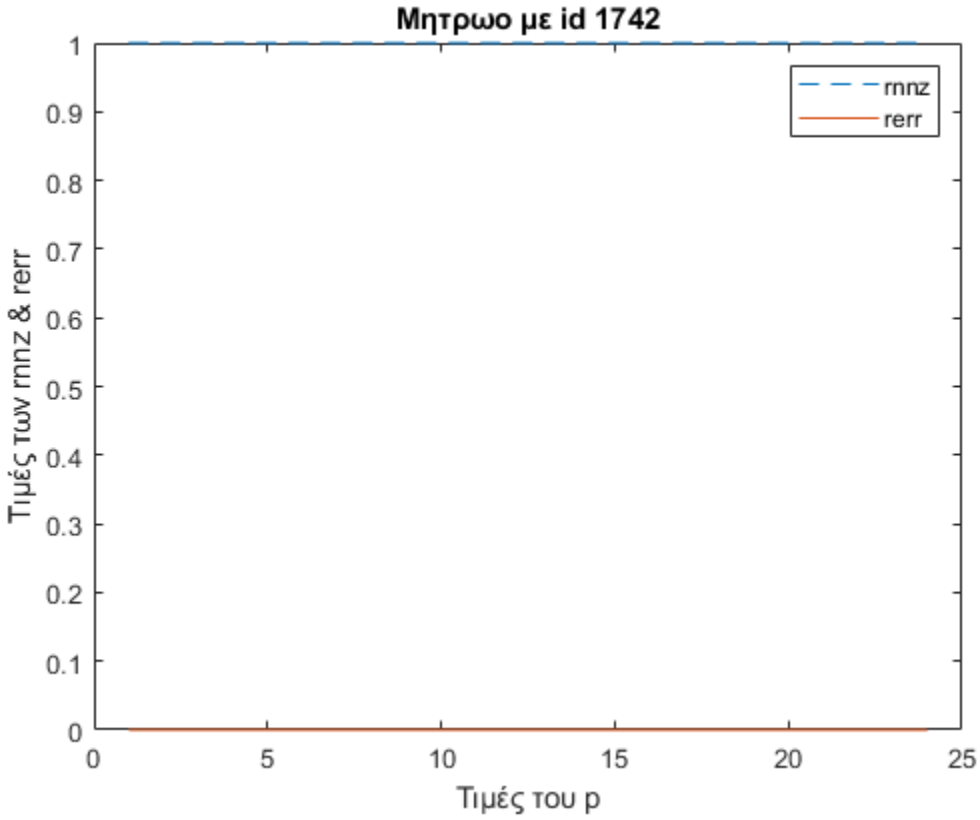
Η συνάρτηση band_stats κατασκευάστηκε ως εξής. Αρχικά, ανάλογα τι είδους μεταβλητή είναι το mxid επιλέγουμε να κατεβάσουμε το σωστό μητρώο από τη συλλογή SuiteSparse ή χρησιμοποιείται το ίδιο το μητρώο mxid.

Εφόσον έχουμε το μητρώο με το οποίο θα δουλέψουμε, κατασκευάζουμε ένα μητρώο C το οποίο περιέχει μόνο τη κύρια διαγώνιο του mxid και υπολογίζουμε τα στατιστικά του. Στη συνέχεια ανοίγουμε ένα βρόχο, ο οποίος κάθε φορά θα προσθέτει μια υποδιαγωνιο από πάνω και από κάτω του ήδη υπάρχοντος μητρώου C. Κάθε φορά θα υπολογίζουμε τα στατιστικά που ζητούνται στην εκφώνηση για όλα τα 'p'.

Ακολουθούν τα αποτελέσματα για τα μητρώα ελέγχου που ζητήθηκαν.




gallery(wathen,10,20)
rnnz
rerr
Τιμές των rnnz & rerr
Τιμές του p
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
100
200
300
400
500
600
700


Μητρωο με id 1742
rnnz
rerr
Τιμές των rnnz & rerr
Τιμές του p
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
5
10
15
20
25

Όπως περιμέναμε οι τιμές και των δυο στατιστικών βρίσκονται στο διάστημα [0 1]. Επίσης το rnnz αυξάνεται συνεχώς μέχρι να φτάσει το 1, ενώ το rerr μειώνεται συνεχώς μέχρι το 0.

Το μητρώο που αντιστοιχεί στο id ίσο με 1742=(1+mod(1741, 2892)), είναι ένα 23*23 ταυτοτικο μητρώο οπότε και τα δυο στατιστικά έχουν τη τιμή 1 καθώς όλες οι τιμές είναι στη διαγώνιο και όλα τα άλλα 0.

5 Επαναληπτικές Μέθοδοι

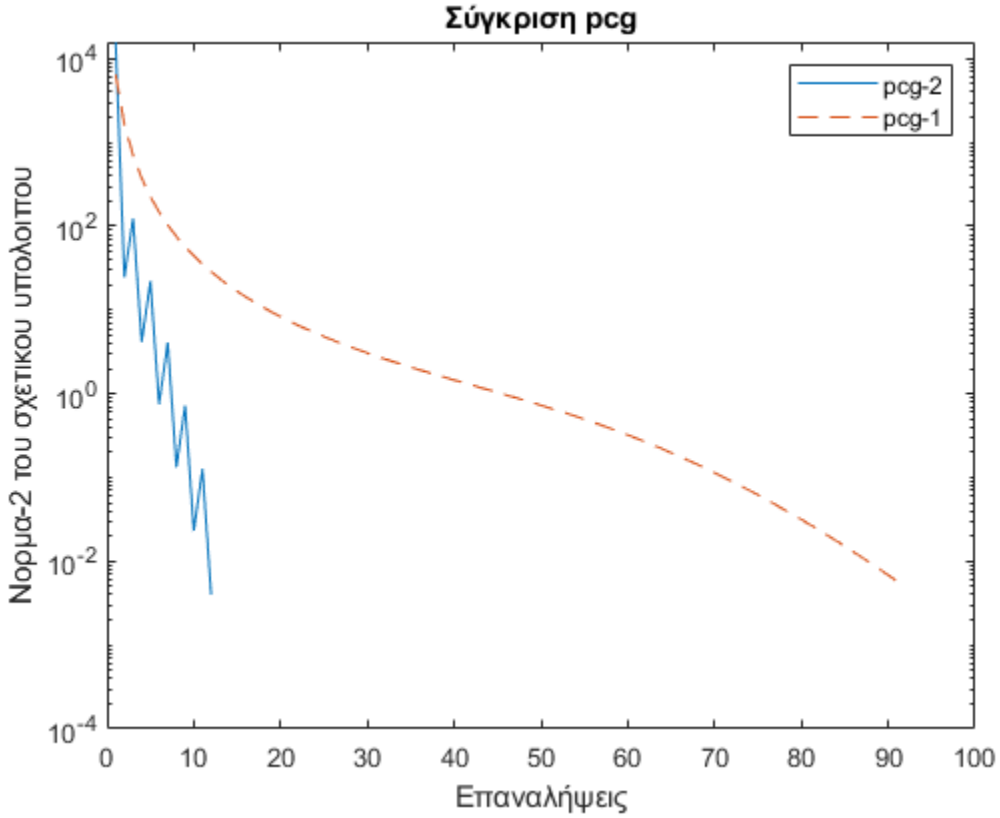
5.1 Ειδικά Μητρώα

5.1.1 PCG-1

Δοκιμάζοντας τη μέθοδο pcg στο συγκεκριμένο μητρώο παρατηρούμε τα εξής. Αρχικά όπως φαίνεται και στο διάγραμμα αλλά και στο FLAG που επιστρέφει η pcg, η μέθοδος συγκλίνει στην 90στη επανάληψη . Όσο για το χρόνο, χρειάστηκε 0.01 δευτερόλεπτα και όσον αφορά την ακρίβεια επέστρεψε σχετικό υπόλοιπο ίσο με $9 * 10^{-7}$. Για το πολλαπλασιασμό Matrix-Vector γνωρίζουμε ότι στη pcg γίνεται μια τέτοια πράξη σε κάθε επανάληψη. Οπότε εφόσον χρειάστηκαν 90 επαναλήψεις για να συγκλίνει, συν το πρώτο M-V για το r_0, έχουμε 91 M-V πολλαπλασιασμούς.

5.1.2 PCG-2

Για το δεύτερο μητρώο έχουμε τα εξής αποτελέσματα. Η pcg συγκλίνει σε 11 επαναλήψεις. Το κάνει αυτό μέσα σε 0.01 δευτερόλεπτα και επιστρέφει σχετικό υπόλοιπο $2.5 * 10^{-7}$. Εφόσον έχουμε 11 επαναλήψεις, έχουμε και (11+1) = 12 M-V πολλαπλασιασμούς.



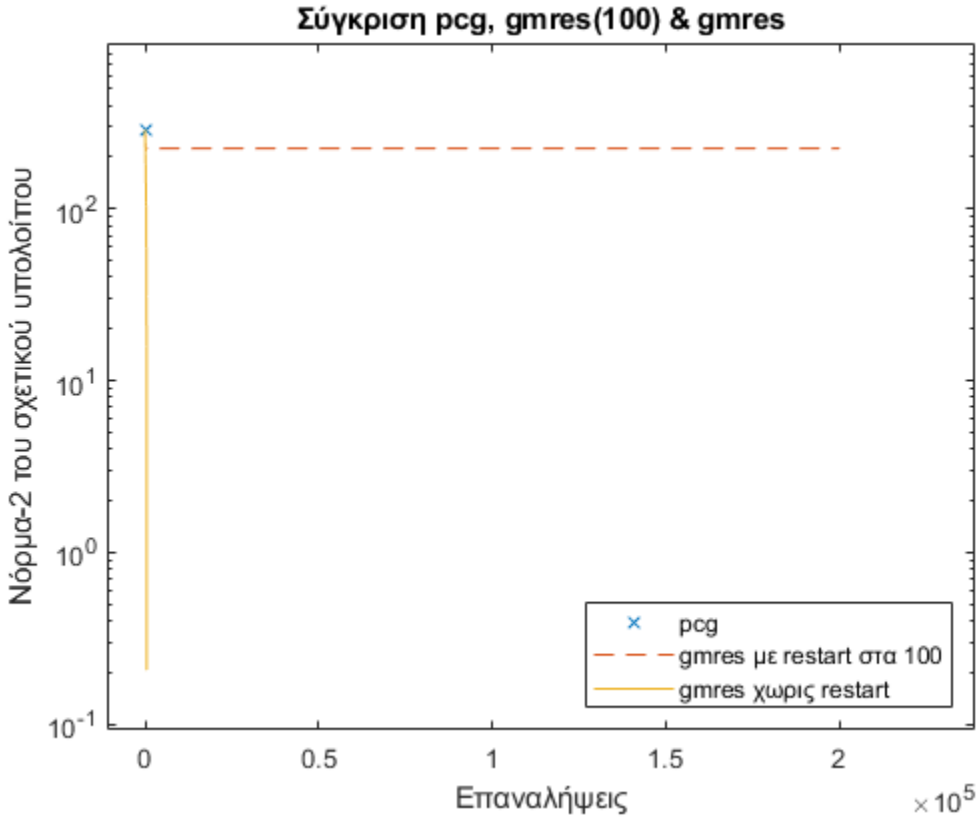
Η νορμα-2 του σφάλματος στο τελευταίο βήμα για το 1ο μητρώο είναι 0.0058, ενώ για το 2ο μητρώο είναι 0.0040

5.1.3 ΣΥΓΚΡΙΣΗ

Συγκρίνοντας τα 2 αποτελέσματα συμπεραίνουμε το εξής. Το δεύτερο μητρώο συγκλίνει πολύ γρηγορότερα από το πρώτο αφού κάνει 79 λιγότερες επαναλήψεις. Επίσης το δεύτερο επιστρέφει πιο ακριβές αποτέλεσμα για τη λύση του Ax=b. Συνεπώς έχουμε και λιγότερους πολλαπλασιασμούς M-V λόγω της διαφοράς στις επαναλήψεις. Το περίεργο που παρατηρούμε αφορά τον δείκτη κατάστασης και είναι το εξής:

Και τα δυο μητρώα είναι διαγώνια, αρά μπορούμε να βρούμε το δείκτη κατάστασης με το τύπο κ(A)=λμαξ/λμιν. Αρά για το πρώτο μητρώο έχουμε κ(A) = 500 και για το δεύτερο κ(A) = 1001. Από τη θεωρία γνωρίζουμε ότι όσο μεγαλύτερος είναι ο δείκτης κατάστασης, τόσο χειρότερη η απόδοση. Εδώ βλέπουμε το αντίθετο. Το δεύτερο έχει διπλάσιο δείκτη κατάστασης αλλά συγκλίνει 9 φορές γρηγορότερα.

5.2 Τυχαία Μητρώα



Η νορμα-2 του σφάλματος στο τελευταίο βήμα για το 1ο μητρώο είναι 224.8, για το 2ο μητρώο είναι 0 και για το 3ο μητρώο είναι 287.6

5.2.1 GMRES χωρίς Restart

Η gmres χωρίς restart συγκλίνει στη 500στη επανάληψη και επιστρέφει λύση που έχει σχετικό υπόλοιπο 0, μέσα σε 0.5 δευτερόλεπτα. Όσον αφορά τους πολλαπλασιασμούς M-V, όπως και στη pcg έτσι και εδώ έχουμε μια τέτοια πράξη σε κάθε επανάληψη (πολλαπλασιάζοντας το A με το Q στη συνάρτηση Arnoldi για να βρεθεί η ορθοκανονικη βάση) - συν ένα για το r_0. Οπότε έχουμε 501 M-V πολλαπλασιασμούς.

5.2.2 GMRES με Restart

Σε αυτή τη περίπτωση δεν έχουμε σύγκλιση. Όπως φαίνεται και στην εικόνα 5.2, η νορμα-2 του υπολοίπου ξεκινάει από τα 287 και στις πρώτες εκατό επαναλήψεις πέφτει στα 226. Από κει και πέρα βελτιώνεται απειροελάχιστα οπότε βλέπουμε μια ευθεία γραμμή στην εικόνα. Όλη η διαδικασία κράτησε 44 δευτερόλεπτα και στο σύνολο είχαμε 200001 πολλαπλασιασμούς M-V. Όσο για το σχετικό υπόλοιπο, αυτό είναι 0.78 στη $200000^{η}$ επανάληψη.

5.2.3 PCG

Με τη χρήση της pcg δε περιμέναμε και πολλά καθώς το μητρώο μας ούτε καν συμμετρικό δεν είναι για να εξετάσουμε αν είναι θετικά ορισμένο. Η κλήση της μεθόδου τερματίζει στη πρώτη επανάληψη με FLAG=4 που σημαίνει ότι κάποιος βαθμωτός έγινε πολύ μικρός ή πολύ μεγάλος για να συνεχίσει. Επιστρέφει σχετικό υπόλοιπο 1 και η νορμα-2 του σφάλματος είναι 287.

Αν δοκιμάσουμε να λύσουμε την εξίσωση με την ανάποδη κάθετο έχουμε τα εξής αποτελέσματα. Αρχικά η λύση βρίσκεται σε 0.008760 δευτερόλεπτα. Όσο για την ακρίβεια έχουμε νόρμα του σχετικού υπολοίπου ιση με $1.2921 * 10^{-14}$ και νόρμα του σχετικού σφάλματος $3.7163 * 10^{-12}$. Η μέθοδος αυτή είναι πολύ ταχύτερη από όλες τις προηγούμενες που προσπαθήσαμε. Όσον αφορά τα αποτελέσματα όμως υπάρχει μια μικρή διαφορά με την gmres χωρίς restart, αλλά ασήμαντη καθώς ο χρόνος είναι πολλές τάξεις μικρότερος.

ΤΕΛΟΣ ΑΝΑΦΟΡΑΣ