ΕΠΙΣΤΗΜΟΝΙΚΟΣ ΥΠΟΛΟΓΙΣΜΟΣ
Εργαστηριακή Άσκηση 2019-20

Ζαχαρίας Γεωργόπουλος

AM : 235735

Περιεχόμενα

1 Εισαγωγή

1.1 Στοιχεία υπολογιστικού συστήματος

1.1.1 Πίνακας στοιχείων συστήματος

Χαρακτηριστικά	Answer
Έναρξη/λήξη εργασίας	24/12/19 - 17/1/20
model	
O/S	Windows 10 Home
processor name	Intel Core i5 4200M
processor speed	2.50 GHz
number of processors	1
total # cores	2
total # threads	4
FMA instruction	
L1 cache	64KB Instruction, 64KB Data write-back
L2 cache	(per core) 256KB, write-back
L3 cache	(shared) 3MB, write-back
Gflops/s	12.84
Memory	6GB
Memory Bandwidth	
MATLAB Version	9.7.0.1190202 (R2019b)
BLAS	
LAPACK	

Ο Πίνακας 1.1.1 συμπληρώθηκε με την βοήθεια του προγράμματος cpuz. Σε αυτό το πρόγραμμα στο section για την CPU βρήκα από το Specification το όνομα του επεξεργαστή, την ταχύτητά του, καθώς και τον αριθμό των cores και των threads. Στο section για τις Caches βρήκα για κάθε μια από τις τρεις cache τον διαμιρασμό τους στα cores και threads της CPU, καθώς και το Size τους. Τέλος, από τις πληροφορίες του PC βρήκα το O/S καθώς και το size της μνήμης.

1.1.2 Έκδοση του MATLAB και σχετικές βιβλιοθήκες

- Matlab version: 9.7.0.1190202 (R2019b)
- Σχετικές βιβλιοθήκες: Figure 1

Εκτέλεση της εντολής "version" στο command window της Matlab και εμφάνιση της έκδοσης της και των σχετικών της βιβλιοθηκών.

1.1.3 Εκτέλεση εντολής "bench"

Πίνακας έπειτα από την εκτέλεση της εντολής "bench": Figure 2

```
>> ver
-----------------------------------------------------------------------------------------------------
MATLAB Version: 9.7.0.1190202 (R2019b)
MATLAB License Number: 612522
Operating System: Microsoft Windows 10 Home Version 10.0 (Build 18362)
Java Version: Java 1.8.0_202-b08 with Oracle Corporation Java HotSpot(TM) 64-Bit Server VM mixed mode
-----------------------------------------------------------------------------------------------------
MATLAB                                                Version 9.7         (R2019b)
Simulink                                              Version 10.0        (R2019b)
Bioinformatics Toolbox                                Version 4.13        (R2019b)
Communications Toolbox                                Version 7.2         (R2019b)
Computer Vision Toolbox                               Version 9.1         (R2019b)
Control System Toolbox                                Version 10.7        (R2019b)
Curve Fitting Toolbox                                 Version 3.5.10      (R2019b)
DSP System Toolbox                                    Version 9.9         (R2019b)
Data Acquisition Toolbox                              Version 4.0.1       (R2019b)
Database Toolbox                                      Version 9.2         (R2019b)
Deep Learning Toolbox                                 Version 13.0        (R2019b)
Embedded Coder                                        Version 7.3         (R2019b)
Fixed-Point Designer                                  Version 6.4         (R2019b)
Fuzzy Logic Toolbox                                   Version 2.6         (R2019b)
GPU Coder                                             Version 1.4         (R2019b)
Global Optimization Toolbox                           Version 4.2         (R2019b)
Image Acquisition Toolbox                             Version 6.1         (R2019b)
Image Processing Toolbox                              Version 11.0        (R2019b)
Instrument Control Toolbox                            Version 4.1         (R2019b)
MATLAB Coder                                          Version 4.3         (R2019b)
MATLAB Compiler                                       Version 7.1         (R2019b)
MATLAB Compiler SDK                                   Version 6.7         (R2019b)
Mapping Toolbox                                       Version 4.9         (R2019b)
Model Predictive Control Toolbox                      Version 6.3.1       (R2019b)
Optimization Toolbox                                  Version 8.4         (R2019b)
Parallel Computing Toolbox                            Version 7.1         (R2019b)
Partial Differential Equation Toolbox                 Version 3.3         (R2019b)

Robust Control Toolbox                                Version 6.7         (R2019b)
Signal Processing Toolbox                             Version 8.3         (R2019b)
SimEvents                                             Version 5.7         (R2019b)
Simscape                                              Version 4.7         (R2019b)
Simscape Electrical                                   Version 7.2         (R2019b)
Simscape Multibody                                    Version 7.0         (R2019b)
Simulink Coder                                        Version 9.2         (R2019b)
Simulink Real-Time                                    Version 6.11        (R2019b)
Stateflow                                             Version 10.1        (R2019b)
Statistics and Machine Learning Toolbox               Version 11.6        (R2019b)
Symbolic Math Toolbox                                 Version 8.4         (R2019b)
System Identification Toolbox                         Version 9.11        (R2019b)
Wavelet Toolbox                                       Version 5.3         (R2019b)
```

Figure 1: Σχετικές Βιβλιοθήκες: (Εντολή "ver" στην Matlab)

Computer Type	LU	FFT	ODE	Sparse	2-D	3-D
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
This machine	0.2620	0.2196	0.0343	0.1756	0.8725	0.7409
MacBook Pro, macOS 10.14.1, Intel Core i5 @ 2.6 GHz	0.2734	0.1980	0.0177	0.1423	2.0198	1.3889

Figure 2: Πίνακας έπειτα από την εκτέλεση εντολής "bench" στην Matlab

1.2 Κατασκευή εργαλείων

1.2.1 Μητρώο μάσκα: Συνάρτηση mask_band

Η συνάρτηση αυτή έχει δύο λειτουργίες. Η 1η είναι όταν το type == "band" και η 2η όταν type == "btdr". Αρχικά, στον κώδικα μου δημιουργώ μητρώο P μεγέθους nxn με άσους στην διαγώνια. Στην 1η περίπτωση ελέγχω αν δόθηκε όρισμα q (αν όχι τότε q=p) και στην συνέχεια με for-loops (με εμφώλευση) προσπαθώ να κατασκευάσω το άνω εύρος του μητρώου P, με βάση το q, δηλαδή δημιουργώ q υπερδιαγωνίους με την τιμή των στοιχείων να είναι 1.Έπειτα, πάλι με for-loops προσπαθώ να κατασκευάσω το κάτω εύρος του μητρώου P, με βάση το p, δηλαδή δημιουργώ p υποδιαγωνίους με την τιμή των στοιχείων να είναι 1. Τώρα, στην 2η λειτουργία ελέγχω αν το p διαιρεί ακριβώς το n (αν όχι θα υπάρχουν k διαγώνια block μεγέθους rxr) και στην συνέχεια δημιουργώ διαγώνια block μεγέθους pxp με άσσους. Με βάση το p φτιάχνω με for-loops και τα αντίστοιχα υπερδιαγώνια και υποδιαγώνια block με άσσους. Στο τέλος λόγω των ορίων επανάληψεις των for-loops, το μητρώ P που δημιουργούταν είχε μέγεθος μεγαλύτερο από nxn, γι'αυτό με την αντίστοιχη εντολή κρατάμα μόνο n γραμμές και n στήλες.

1.2.2 Έλεγχος Διαγώνιας Κυριαρχίας: Συνάρτηση dd_check

Για την συνάρτηση αυτή ορίζω στην αρχή διάνυσμα a το οποίο περιέχει κατά απόλυτη τιμή τα διαγώνια στοιχεία του μητρώου A. Στην συνέχεια, ελέγχω την Διαγώνια Κυριαρχία κατά γραμμές με ένα for-loop το οποίο αθροίζει τα στοιχεία της i-οστής γραμμής κάθε φορά. Έπειτα με χρήση if-else statement ελέγχω αν είναι ΔΚ κατά γραμμές και αν είναι Αυστηρά ΔΚ κατά γραμμές. Αν δεν είναι, θέτω την μεταβλητή adkR = 1. Στην συνέχεια, ακολουθώ την ίδια διαδικασία για τις στήλες, κάνοντας του αντίστοιχους ελέγχους και δίνοντας στις μεταβλητές τις αντίστοιχες τιμές. Γενικά, το discrC (για στήλες) και το discrR (για γραμμές) αυξάνεται κάθε φορά που μια στήλη ή γραμμμή αντίστοιχα επιτυγχάνει την ΔΚ. Στο τέλος, ελέγχω αν το μητρώο είναι ΔΚ κατά γραμμές και στήλες (αν ναι κάνω το dflag = 1). Επίσης, ελέγχω αν το μητρώο είναι ΑΔΚ κατα γραμμές (αν όχι discrR = 0) και ΑΔΚ κατά στήλες (αν οχι discrC = 0).

2 Πράξεις με μητρώα ειδικής μορφής, πολυώνυμα μητρώων και χρήση διεπαφής mex

2.1 Μητρώα και δεξιά μέλη

Το αρχείο cmatrix.m περιλαμβάνει την δημιουργία των μητρών που μας ζητείται στην εκφώνηση, τα οποία είναι C, P1, P2, P3, M1, M2. Το μητρώο C δημιουργήθηκε με χρήση δύο for-loop και ένα if-else statement με το οποίο δίνω τις αντίστοιχες τιμές στα διαγώνια στοιχεία και σε αυτά εκτός διαγωνίου. Τα μητρώα P1, P2, P3 έχουν διαφορετικό size το καθένα (m,n). Δημιουργώ τα μητρώα T και S αντίστοιχα κάθε φορά κάνοντας χρήση της συνάρτησης toeplitz που μας δίνετε. Τέλος, βρίσκω τα αραιά (sparse) μητρώα εκτελόντας τις πράξεις kron, μεταξύ των αντίστοιχων μητρώων.

Για να φτιάξω τον παρακάτω πίνακα στην αρχή εκτέλεσα το αρχείο cmatrix.m για να δημιουργήσει τα αντίστοια μητρώα. Έπειτα για κάθε ένα από τα μητρώα έκανα τα εξής: Το N συμβολίζει το μέγεθος του μητρώου (έστω M) και το βρίσκω με την εντολή length(M). Για την ΔΚ εκτελώ για το κάθε μητρώο την συνάρτηση dd_check(M) και αν dflag = 1, τότε είναι ΔΚ. Ένα μητρώο είναι συμμετρικό αν (M*M'=I) η εντολή disp(isequal(M,M.') δώσει σαν αποτέλεσμα 1, αλλιώς δίνει 0. Το ζώνης το υπολογίζω με χρήση της συνάρτησης [lower,upper] = bandwidth(M), η οποία βρίσκει το κάτω και άνω εύρος του μητρώου M. Ελέγχω αν το μητρώο είναι αντιστρέψιμο με την εντολή det(M) και αν το αποτέλεσμα είναι διάφορο του 0 τότε είναι αντιστρέψιμο, αλλιώς όχι. Τέλος, βρίσκω τον δείκτη κατάστασης ως προς τη νόρμα 1 εκτελώντας την εντολή condest(M).

Πίνακας χαρακτηριστικών μητρώων ελέγχου.

matrix	N	ΔΚ	Συμμετρικό	Ζώνης	Αντιστρέψιμο	Δ.Κ. k1(A)
toeplitz ([2,-1, zeros(1,8)])	10	Ναι	Ναι	(1,1)	Ναι	60
C1000	1000	Όχι	Ναι	(999,999)	Ναι	41.2165
P(100,10)	1000	Όχι	Ναι	(100,100)	Ναι	139.9998
P(10,100)	1000	Όχι	Ναι	(10,10)	Ναι	135.6683
P(100,100)	1000	Όχι	Ναι	(100,100)	Ναι	7.0018e+03
bcsstm07	420	Όχι	Ναι	(47,47)	Ναι	1.3365e+04
email	1133	Όχι	Ναι	(605,605)	Όχι	Inf

2.2 Επίλυση με απλά πολυώνυμα μητρώου

Αρχικά, επειδή μας ζητείται η πράξη να εκτελεστεί αποκλειστικά μέσω BLAS-2, τότε είναι ανέφικτο να πραγματοποιήσουμε την πράξη A*A (μητρώο * μητρώο, BLAS-3), αφού όταν έχουμε ύψωση μητρώου σε δύναμη είναι σαν να πολλαπλασιάζουμε το μητρώο αυτό (i-1)-φορές με τον εαυτό του, οπότε θα πρέπει να καταφύγουμε σε άλλη λύση, όπως είναι ο πολλαπλασιασμός μητρώου με διάνυσμα. Η συνάρτηση θέλουμε να υπολογίζει το p(A)*b με BLAS-2, γι'αυτό θα υπολογίζουμε πρώτα το p(i)*b (το οποίο δίνει σαν αποτέλεσμα ένα διάνυσμα χ) και στην συνέχεια (για το αντίστοιχο i) θα πραγματοποιούμε την πράξη A*u (μητρώο*διάνυσμα = διάνυσμα x) επαναληπτικά όσες φορές "δείχνει" η δύναμη του μητρώο A, στην οποία θέλουμε να υψωθεί.

Επί της ουσίας θέλουμε να υπολογίσουμε το άθροισμα των παρακάτω πράξεων, το οποίο θα μας δώσει στο τέλος το διάνυσμα r.

$$p(A)*b = \begin{cases} A^4 * p(1) * b, & [i=1, j=1:n-1] \\ A^3 * p(2) * b, & [i=2, j=1:n-2] \\ A^2 * p(3) * b, & [i=3, j=1:n-3] \\ A * p(4) * b, & [i=4, j=1:n-4] \\ p(5) * b, & [i=5] \end{cases}$$

Ο αλγόριθμος (κώδικας) υλοποίησης του παραπάνω λειτουργεί ως εξής. Με ένα for-loop υπολογίζουμε κάθε φορά το x= p(i)*b και στην συνέχεια με ένα εμφωλευμένο for-loop πραγματοποιούμε j-φορές την πράξη x=A*x. Κάθε ένα αποτέλεσμα που βρίσκω μετά το εμωλευμένο for-loop το προσθέτω με το προηγούμενο αποτέλεσμα. Αν το i γίνει κάποια στιγμή ίσο με το n, τότε βγαίνουμε έξω από το for-loop,προσθέτωντας το x=p(n)*b (τελευταίος όρος του πολυωνύμου) με το προηγούμενο άθροιμσα.

2.2.1 Διαδικασίες επίλυσης

Το αρχείο script.m περιέχει τον explicit υπολογσμό του C=p(A) και την επίλυση του C*x=b με την ανάποδη κάθετο της Matlab. Ακομά, περιλαμβάνει τον serial υπολογισμό των ριζών του p(z), καθώς και την επίλυση του παρακάτω συστήματος, με τις μεθόδους της ανάποδης καθέτου, της pcg χωρίς προρρύθμιση και με προρρύθμιση.

$$(A - \zeta_j * I) * x^{(j)} = x^{(j-1)}, \ \ \mu\varepsilon \ \ x^{(0)} = b \ \ \kappa\alpha\iota \ \ x = x^{(d)} \ \ \kappa\alpha\iota \ \ j = 1, ..., d.$$

Τώρα, για την **Explicit** επίλυση στο αρχείο script.m υπολογίζω το C=p(A) με χρήση της συνάρτησης polyvalm(p,A), η οποία υπολογίζει το εξής:

$$C = p(A) = p(1) * A^n + ... + p(i-1) * A^{n-1} + p(n)$$

.

Επίσης, υπολογίζω το C*x=b με την ανάποδο κάθετο, δηλαδή x=C\b (συνήθως η Matlab το υπολογίζει με LU παραγοντοποίηση).

Για την **Serial** επίλυση και τον υπολογισμό των ριζών του p(z) χρησιμοποιώ την εντολή root() της Matlab η οποία βρίσκει τις ρίζες του πολυωνύμου. Μετά, με τη χρήση της ανάποδης καθέτου και ενός for-loop πραγματοποιώ την παρακάτω πράξη

$$(A - \zeta_j * I) * x^{(j)} = x^{(j-1)}, \ \ => x^{(j)} = (A - \zeta_j * I) \backslash x^{(j-1)}$$

για καθένα από τα j (από 1 έως το πλήθος των ριζών του p). Έπειτα, για τον υπολογισμό του συστήματος με τη χρήση της pcg χωρίς προρρύθμιση, πάλι με ένα for-loop (και αντίστοιχες τιμές του j) αρχικά υπολογίζω το μητρώο

P1=(A - r(l)*I), με l να είναι το length των ριζών του p(z) μείον την τρέχουσα τιμή του j και ένα. Δηλαδή, έχουμε:

$$P1 = \begin{cases} A - r(l) * I, & j = 1, l = length(r) \\ A - r(l) * I, & j = 2, l = length(r) - 1 \\ . \\ . \\ . \\ A - r(1) * I, & j = length(r), l = 1 \end{cases}$$

Η pcg(P1,y) (με y=b) προσπαθεί να λύσει το γραμμικό σύστημα P1*p1=y. Το μητρώο P1 πρέπει να είναι συμμετρικό, θετικά ορισμένο, μεγάλο και αραιό, ενώ το διάνυσμα y πρέπει να έχει length όσο το μητρώο P1. Το όρισμα tol ορίζει το max σφάλμα για το οποίο θεωρείται η μέθοδος επιτυχής. Το όρισμα maxit ορίζει τον μέγιστο αριθμό επαναλήψεων για να επιτύχουμε το σφάλμα μικρότερο του tol.

Τέλος, για τον υπολογισμό του συστήματος με τη χρήση της pcg με προρρύθμιση Block Jacobi, δημιουργώ στην αρχή ένα μηδενικό μητρώο M με μέγεθος όσο το μητρώο C και με δύο for-loops (και μια συνθήκη if) γεμίζω την διαγώνιο το M με τα στοιχεία της διαγωνίο C. Μετά, με χρήση for-loop υπολογίζω το μητρώο P2 (όπως υπολόγισα το P1) και στην συνέχεια καλώ την pcg με τα εξής ορίσματα (P2,y,tol,maxit,M). Το διαφορετικό σε αυτήν την υλοποίηση είναι το όρισμα M, το οποίο είναι ένα συμμετρικα θετικά ορισμένο μητρώο, το οποίο επιλύει το σύστημα inv(M)*P2*p2 = inv(M)*y, ως προς p2.

Τώρα, όσον αφορά την χρονομέτρηση κάθε επίλυσης χρησιμοποιούμε της εντολές tic (για έναρξη της χρονομέτρησης) και toc (για λήξη της χρονομέτρησης). Για να είναι πιο ορθά τα αποτελέσματα της χρονομέτρης, ετκελώ πολλές φορές την μέθοδο επίλυσης μου (με χρήση for-loop) και στο τέλος παίρνω τον μέσο όρο των χρονομετρήσεων. Για να υπολογίσω τους χρόνους εκτέλεσης σε second, εκτελώ κάθε φορα το script.m, με το αντίστοιχο μητρώο, π.χ. αν θέλω να βρω το χρόνο εκτέλεσης του C_1000 για τα συστήματα που αναφέρθηκαν παραπάνω, τότε στην 14η γραμμή αναθέτω στο μητρώο A την τιμή του C (το έχω υπολογίσει στο αρχείο cmatrix.m). Στο command window της Matlab καλώ τους αντίστοιχους χρόνους εκτέλεσης για κάθε σύστημα επίλυσης, οι οποίοι είναι οι t1,t2,t3,t4. Αυτό το κάνω και για τα 5 μητρώα του παρακάτω πίνακα.

Πίνακας χαρακτηριστικών μητρώων ελέγχου.

RUNTIMES (sec)	C_{1000}	$P_{(10,100)}$	$P_{(100,10)}$	$P_{(100,100)}$	bcsstm07
explicit	5.7000e-03	5.0000e-03	6.2000e-03	7.4627e+00	6.3400e-02
serial+backslash	2.5500e-02	2.2600e-02	2.8900e-02	3.1603e+01	3.9100e-02
serial+Cholesky	X				
serial+PCG	X	1.0400e-02	1.0500e-02	1.2605e+01	5.2400e-02
serial+PCG+prec(blockJ)	X	6.3000e-02	6.5700e-02	2.4646e+01	1.1800e-01
parallel+backslash (προ-2014)					

Τώρα, όσον αφορά τις νόρμες σφαλμάτων χρησιμοποιώ την έτοιμη συνάρτηση της Matlab "norm(result-c,2)", η οποία υπολογίζει την 2η νόρμα του αποτελέσματος της επίλυσης του συστήματος μείον το μοναδιαίο μητρώο c.

Πίνακας: Νόρμες-1 σφαλμάτων $\| e - x \|_2$.

ERRORS ($\| \cdot \|_2$)	C_{1000}	$P_{(10,100)}$	$P_{(100,10)}$	$P_{(100,100)}$	bcsstm07
explicit	5.7780e+07	1.0972e+07	1.9676e+07	1.8446e-09	7.7908e-06
serial+backslash	4.5119e+26	1.5354e+26	1.7584e+26	1.8144e-12	7.1312e-08
serial+Cholesky	X				
serial+PCG	X	1.8996e+39	1.8996e+39	4.8594e+05	2.3145e+09
serial+PCG+prec(blockJ)	X	1.6184e+43	1.6035e+43	1.5304e+06	2.2704e+10
parallel+backslash (προ-2014)					

3 Εφαρμογές σε δίκτυα

Για το ερώτημα αυτό, φορτώνω στην αρχή το μητρώο A του email (το ονομάζω M2), με τις παρακάτω εντολές στο command window της Matlab.

E = load('email.mat','-mat'); και M2 = E.Problem.A;

Τώρα, αν θέλω να πραγματοποιήσω την direct επίλυση, τότε στα ορίσματα της συνάρτησης δεν συμπληρώνω το cell pcg_parms, αν θέλω να πραγματοποιήσω την επίλυση pcg, τότε στα ορίσματα της συνάρτησης συμπληρώνω το cell pcg_parms. Αρχικά, χρησιμοποιώ ένα for-loop 5 επαναλήψεων για να χρονομετρήσω με ακρίβεια το runtime (t) της μεθόδου επίλυσης που θα ακολουθήσει. Με ένα εμφωλευμένω for-loop υπολογίζω το μητρώο M για μια συγκεκριμένη τιμή alpha κάθε φορά (αφού το i παίρνει τιμές από 1 έως το μήκος του διανύσματος alpha). Μετά, πάλι με ένα if statement ελέγχω αν το μητρώο M έχει ορίζουσα διάφορη του μηδέν, το οποίο σημαίνει ότι το M είναι αντιστρέψιμο. Έπειτα, ελέγχω με ένα if-elseif statement αν mth == "direct" ή αν mth == "pcg" και εκτελώ την αντίστοιχη μέθοδο επίλυσης (αν direct τότε υπολογισμός συστήματος με χρήση ανάποδης καθέτου, αν pcg εκτέλεση της pcg με τα αντίστοιχα ορίσματα το αραιό μητρώο M, το διάνυσμα e, το max σφάλμα (tol), για το οποίο είναι αποδεκτή η λύση και το max αριθμό επαναλήψεων (maxit) που μπορούν να πραγματοποιηθούν για να επιτύχουμε σφάλμα μικρότερο του tol και χρήση του output της συνάρτησης που μας δίνει στοιχεία όπως το iter, το οποίο δηλώνει τον αριθμο επαναλήψεων που πραγματοποιήθηκαν). Στην συνέχεια, ελέγχω με την συνάρτηση all() της matlab, αν όλα τα στοιχεία του διανύσματος x είναι θετικά και αν είναι εισάγω στο διάνυσμα valid την έγκυρη τιμή του alpha και στο μητρώο X (αποτέλεσμα της συνάρτησης που ζητείται) βάζω τις τιμές του διανύσματος x στην αντίστοιχη στήλη του μητρώου.

Τώρα, για τον πίνακα που μας ζητείται, βρίσκω τις valid τιμές του alpha, από το διάνυσμα valid. Το runtime το υπολογίζω από τον χρόνο που κατέγραψε το t1 και t2, αντίστοιχα για τις μεθόδους explicit και pcg. Ακόμα, τις επαναλήψεις της pcg τις υπολόγισα μέσω της μεταβλητής iter (output της συνάρτησης pcg). Η νόρμα που μου ζητείται την υπολόγισα ως εξής και για τις δύο μεθόδους επίλυσης. Αρχικά, δήλωσα το b ως μοναδιαίο διάνυσμα με μέγεθος όσο το length του M2, όπου A έβαλα ο μητρώο M2 και στο x έβαλα την στήλη του μητρώου X που αντιστοιχεί στο αντίστοιχο alpha του πίνακα. Η εντολή που έγραψα στο command window της matlab ήταν: norm(b-M2*X(:,i) ,2), όπου i η στήλη του αντίστοιχου alpha.Τέλος, για να βρώ τα top-5 (hi-to-lo), δηλαδή τις θέσεις των top-5 στοιχείων του μητρώου X για ένα συγκεκριμένο alpha, τότε καλώ την συνάρτηση [out,idx]=sort(X(:,i)), η οποία για το συγκεκριμένο alpha (στήλη i) επιστρέφει απο το μικρότερο προς το μεγαλύτερο τις θέσεις των στοιχείων.

Πίνακας ζητούμενων αποτελεσμάτων.

1-alpha(1)*email	a	runtime(sec)	επαναλήψεις	$\parallel b - A * \widehat{x} \parallel_2$	top-5 (hi to lo)
format	x.xx	xx.xxxx	xxx	x.xxxxe$\pm xx$	π.χ. (1133,1029,2,1,15)
explicit	0.01	00.1047	x	5.3780e+02	(105,333,42,16,23)
	0.04	00.1097	x	2.4501e+03	(105,16,42,196,333)
serial+PCG	0.01	00.0755	7	5.3780e+02	(105,333,42,16,23)
	0.05	00.0615	1	8.4604e+02	(1133,1132,1131,1130,1129)
serial+PCG+prec(ichol)	a_{min}				
	a_{max}				

ΤΕΛΟΣ ΑΝΑΦΟΡΑΣ

(Η αναφορά γράφτηκε σε LaTex, με χρήση του Overleaf.)