

ΕΠΙΣΤΗΜΟΝΙΚΟΣ ΥΠΟΛΟΓΙΣΜΟΣ

Άσκηση 2019-20



CEID
COMPUTER ENGINEERING & INFORMATICS DEPARTMENT

Πάυλος Βραχνής

AM: 26010

Περιεχόμενα

| | |
|-----------|----|
| Ερώτημα 1 | 2 |
| 1.1 | 2 |
| 1.2 | 4 |
| 1.3 | 5 |
| 1.4 | 6 |
| Ερώτημα 3 | 9 |
| 3.1 | 9 |
| 3.2 | 10 |
| 3.2.1 | 11 |
| Ερώτημα 4 | 14 |

Υποσημείωση: Αρκετά αρχεία έχουν μέσα και κώδικα είτε για αποτελέσματα που ζητούνται έπειτα(γιατί δεν γνωρίζω αν επιθυμείτε να στείλουμε επιπλέον αρχεία) είτε για να δημιουργούνται μητρώα που ζητούνται στην συνέχεια και να μην δίνονται από το command window. Σε κάθε περίπτωση υπάρχει επεξήγηση στα σχόλια.

Σε πολλές συναρτήσεις έχω σε σχόλιο έτοιμες εντολές για να ελέγξετε την ορθότητα.

Ερώτημα 1

1.1

i)

Τα παρακάτω στοιχεία βρέθηκαν με το πρόγραμμα CPUID CPU-Z, με το System Information από το toolbox του Advanced System care, όπως με με τον πίνακα του System Properties των Windows 7.

Πίνακας 1: Στοιχεία για τα πειράματα

| Χαρακτηριστικό | Απάντηση |
|----------------------|---|
| Έναρξη/λήξη εργασίας | 02/01/19 - 18/1/20 |
| model | Custom Desktop |
| O/S | Microsoft Windows 7 Ultimate 6.1.7601 |
| processor name | Intel Core i5 7600 |
| processor speed | 3.50 GHz |
| number of processors | 1 |
| total # cores | 4 |
| total # threads | 4 |
| FMA instruction | - |
| L1 cache | 256KB Instruction, 64KB Data write-back |
| L2 cache | (per core) 1024KB, write-back |
| L3 cache | (shared) 6MB, write-back |
| Gflops/s | 280.7 |
| Memory | 16GB |
| Memory Bandwidth | 35.76GB/s |
| BLAS | - |
| LAPACK | - |
| MATLAB Version | 9.4.0.813654 (R20184) |

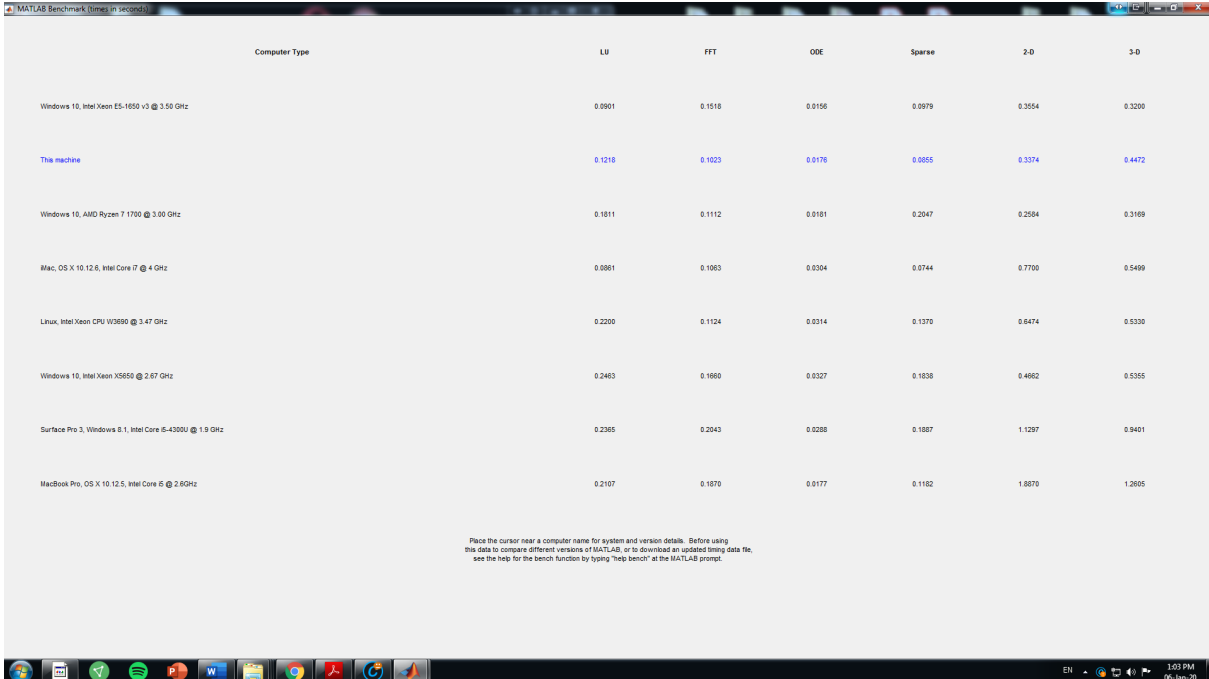
ii)

Έκδοση του MATLAB: '9.4.0.813654 (R2018a)'

Το αποτέλεσμα το πήρα από την εκτέλεση της εντολής version

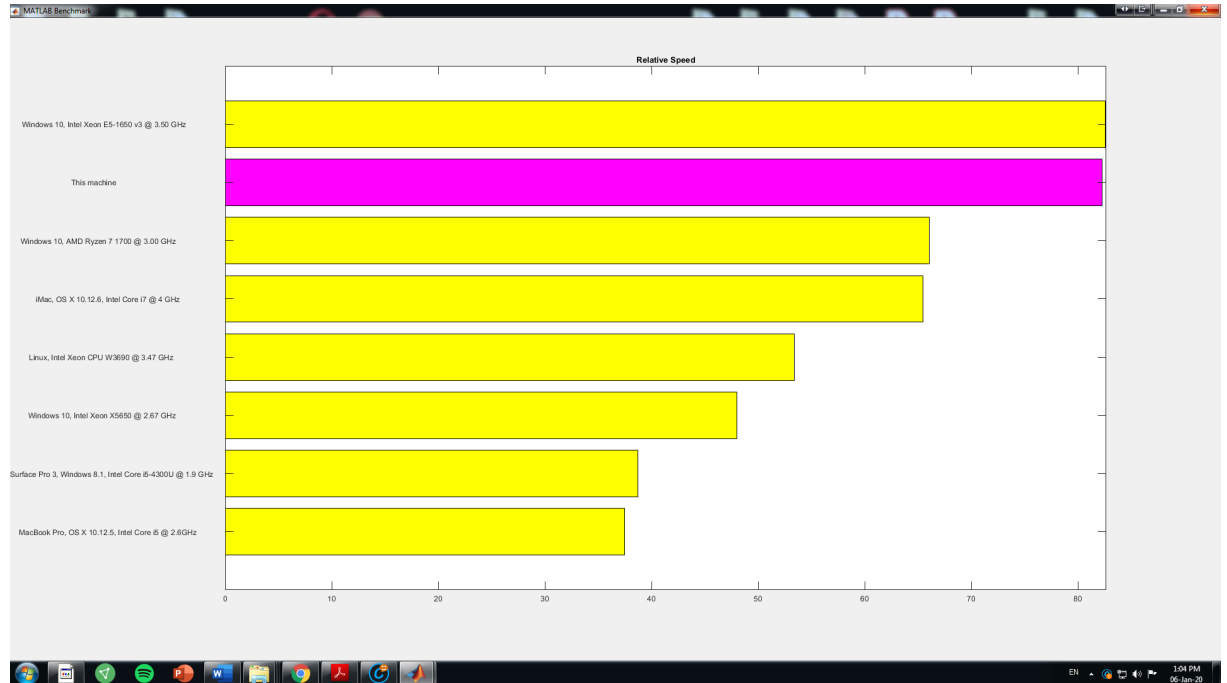
iii)

Αποτελέσματα από την εκτέλεση της εντολής bench



| Computer Type | LU | FFT | ODE | Sparse | 2-D | 3-D |
|--|--------|--------|--------|--------|--------|--------|
| Windows 10, Intel Xeon ES-1650 v2 @ 3.50 GHz | 0.0901 | 0.1518 | 0.0156 | 0.0979 | 0.3554 | 0.3200 |
| This machine | 0.1218 | 0.1023 | 0.0176 | 0.0655 | 0.3374 | 0.4472 |
| Windows 10, AMD Ryzen 7 1700 @ 3.00 GHz | 0.1811 | 0.1112 | 0.0181 | 0.2047 | 0.2584 | 0.3169 |
| Mac, OS X 10.12.6, Intel Core i7 @ 4 GHz | 0.0881 | 0.1063 | 0.0304 | 0.0744 | 0.7700 | 0.5499 |
| Linux, Intel Xeon CPU W3690 @ 3.47 GHz | 0.2200 | 0.1124 | 0.0314 | 0.1370 | 0.6474 | 0.5330 |
| Windows 10, Intel Xeon X5550 @ 2.97 GHz | 0.2463 | 0.1660 | 0.0327 | 0.1638 | 0.4662 | 0.5355 |
| Surface Pro 3, Windows 8.1, Intel Core 5-4300U @ 1.9 GHz | 0.2365 | 0.2043 | 0.0288 | 0.1887 | 1.1287 | 0.9401 |
| MacBook Pro, OS X 10.12.5, Intel Core 6 @ 2.6GHz | 0.2107 | 0.1870 | 0.0177 | 0.1182 | 1.8870 | 1.2605 |

Place the cursor near a computer name for system and version details. Before using this data to compare different versions of MATLAB, or to download an updated testing data file, see the help for the bench function by typing "help bench" at the MATLAB prompt.



1.2

Συνάρτηση mask_band: Αρχικά γίνεται έλεγχος για το όρισμα type .Αν είναι ίσο με 'band', τότε αρχικά κατασκευάζω το μητρώο A με '1' στην κύρια διαγώνιο. Έπειτα βάζω τα υπερδιαγώνια στοιχεία με βάση το q αν η συνάρτηση έχει 3 ορίσματα. Αν έχει 4 τότε βάζω τα υπερδιαγώνια στοιχεία με βάση το όρισμα p. Μετά βάζω τα υποδιαγώνια στοιχεία με βάση το όρισμα p.Αν το type είναι ίσο με 'btdr',τότε αρχικά α ελέγχω αν το p διαιρεί ακριβώς το n. Αν ναι συνεχίζω αλλιώς το στρογγυλοποιώ προς τα κάτω. Στη συνέχεια κατασκευάζω διαγώνια,υπεραδιαγώνια και υποδιαγώνια μπλοκ με '1'. Τέλος κρατάμε μόνο το πίνακα n επί n.

Παραθέτω μία εκτέλεση της εντολής:

```
Command Window
>> mask_band(4,"band",2)
ans =
    1    1    1    0
    1    1    1    1
    1    1    1    1
    0    1    1    1

>> mask_band(5,"band",2)
ans =
    1    1    1    0    0
    1    1    1    1    0
    1    1    1    1    1
    0    1    1    1    1
    0    0    1    1    1

>> mask_band(5,"bodr",3)
ans =
    1    1    1    1    1
    1    1    1    1    1
    1    1    1    1    1
    1    1    1    1    1
    1    1    1    1    1

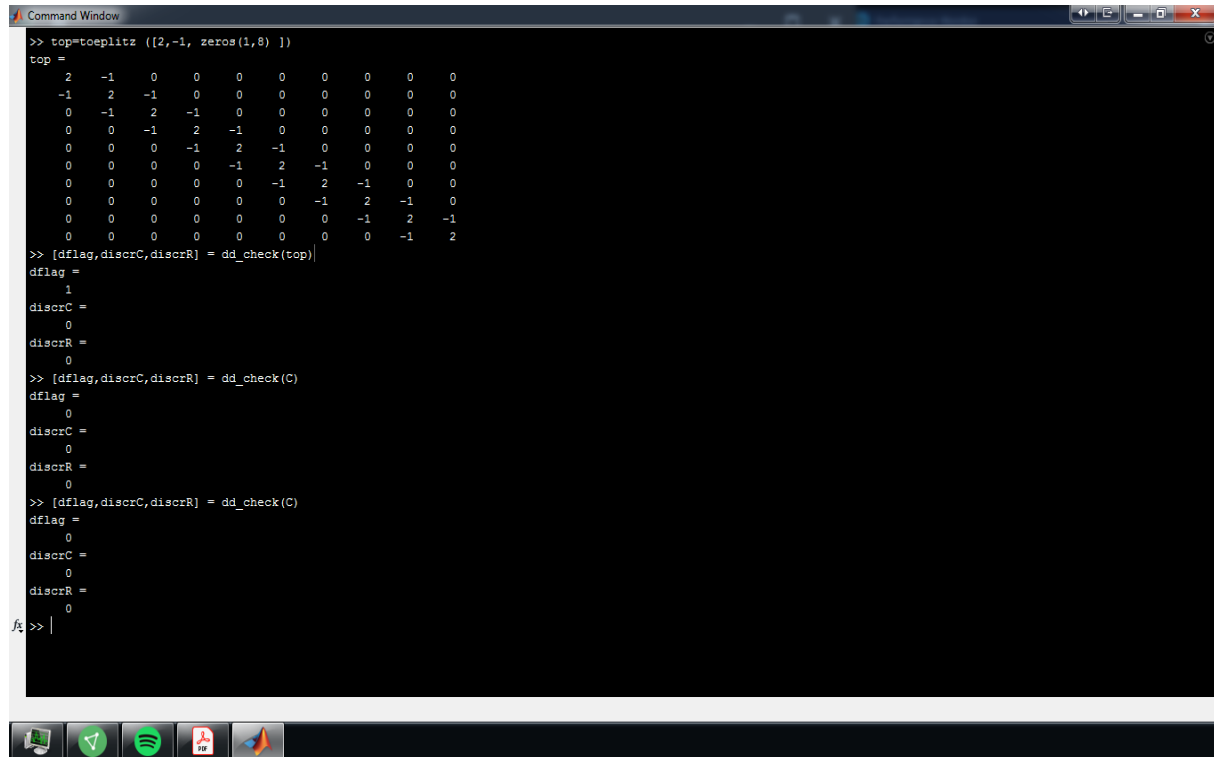
>> mask_band(5,"bodr",2)
ans =
    1    1    1    1    0
    1    1    1    1    0
    1    1    1    1    1
    1    1    1    1    1
    0    0    1    1    1

fx >> |
```

1.3

Συνάρτηση dd_check:(Περιέχει και το ερώτημα για τον Πίνακα 2, που εξηγείται παρακάτω.) Αρχικά βρίσκω το sum των γραμμών χωρίς το στοιχείο της διαγωνίου και εξετάζω για ΑΔΚ και ΔΚ κατά γραμμές. Έπειτα βρίσκω το sum των στηλών χωρίς το στοιχείο της διαγωνίου και εξετάζω για ΑΔΚ και ΔΚ κατά στήλες. Αν δεν είναι ΔΚ τότε το dflag είναι ίσο με '0' αλλιώς με '1'. Αν είναι ΑΔΚ κατά γραμμές το checkR είναι ίσο με '0', αλλιώς με '1' και αντίστοιχα για το checkC. Αν dflag ίσο με '1' τότε αν το checkR ισούται με '0' το discrR γίνεται '0' αλλιώς μένει ως έχει και αντίστοιχα για το discrC. Αν dflag ισούται με '0' τότε τα discrR και discrC μηδενίζονται.

Παραθέτω μία εκτέλεση της εντολής:



```
>> top=toeplitz ([2,-1, zeros(1,8) ])
top =
     2    -1     0     0     0     0     0     0     0     0
    -1     2    -1     0     0     0     0     0     0     0
     0    -1     2    -1     0     0     0     0     0     0
     0     0    -1     2    -1     0     0     0     0     0
     0     0     0    -1     2    -1     0     0     0     0
     0     0     0     0    -1     2    -1     0     0     0
     0     0     0     0     0    -1     2    -1     0     0
     0     0     0     0     0     0    -1     2    -1     0
     0     0     0     0     0     0     0    -1     2    -1
     0     0     0     0     0     0     0     0    -1     2

>> [dflag,discrC,discrR] = dd_check(top)
dflag =
     1
discrC =
     0
discrR =
     0

>> [dflag,discrC,discrR] = dd_check(C)
dflag =
     0
discrC =
     0
discrR =
     0

>> [dflag,discrC,discrR] = dd_check(C)
dflag =
     0
discrC =
     0
discrR =
     0

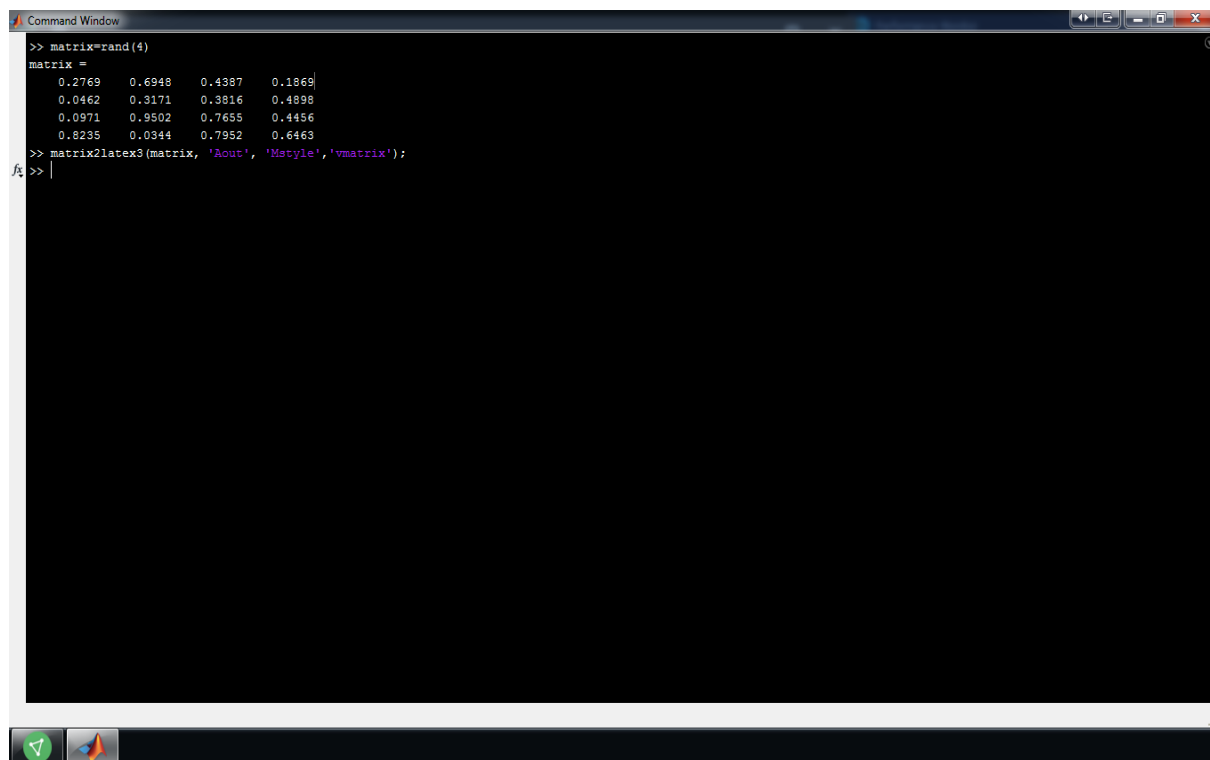
>> |
```

1.4

Στον κώδικα του Koehler πρόσθεσα το Mstyle όπως όλα τα arguments. Δηλαδή το αρχικοποίησα ως [] και πρόσθεσα μία case ακόμα για αυτό. Επίσης πρόσθεσα 3 περιπτώσεις ακόμα για pmatrix, vmatrix και bmatrix. Η λογική στον κώδικα είναι να γράψω αρχικά για το καθένα begin {equation*} Matrix= begin {matrix} και στην συνέχεια για κάθε στήλη να προσθέτω εκτός από την τελευταία και για κάθε γραμμή να αλλάζω σειρά. Επίσης στο τέλος προσθέτω end {matrix} end {equation*} για να τερματίσω.

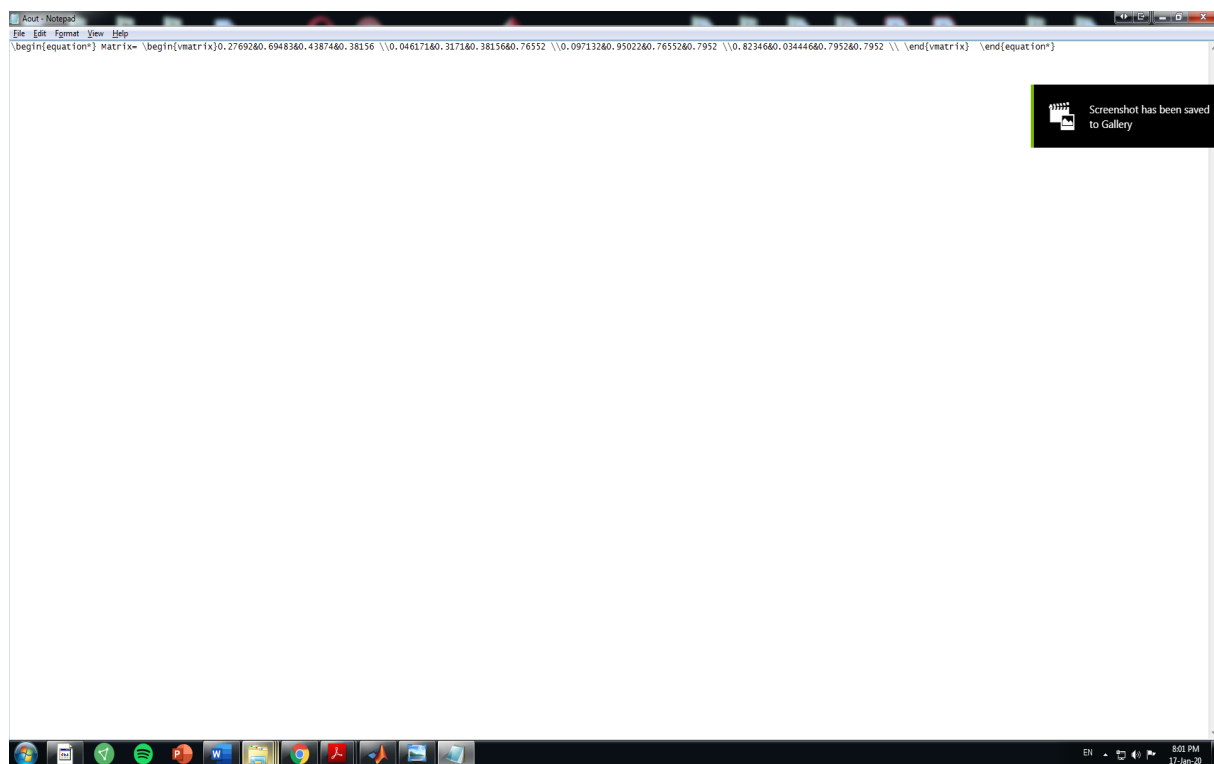
Παραθέτω μία εκτέλεση για ένα τυχαίο μητρώο `matrix=rand(4)` με 3 εικόνες μία για την εκτέλεση, μία για το αποτέλεσμα και μία για την εκτέλεση σε latex:

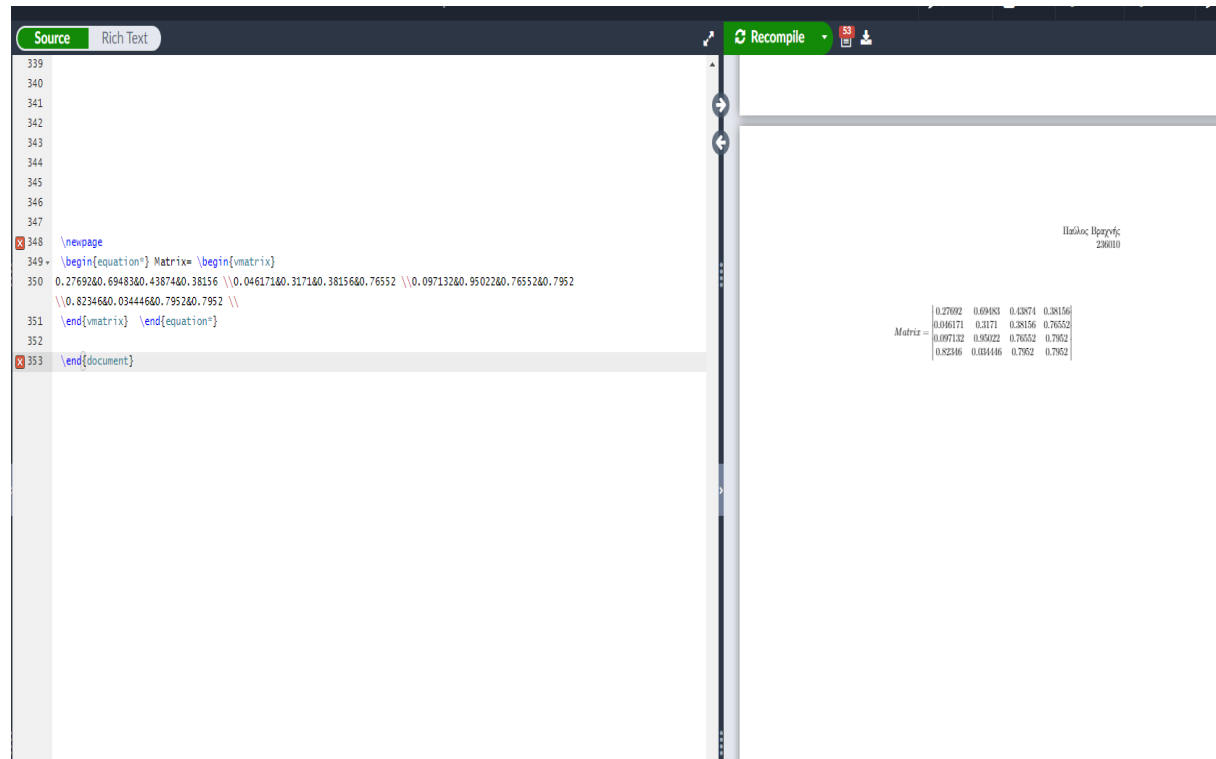
*Η συνάρτηση τρέχει κανονικά και για `matrix2latex2(matrix, 'Aout', 'Mstyle','bmatrix','format','%4.2f');`

A screenshot of the MATLAB Command Window. The window title is "Command Window". The command prompt shows the execution of `matrix=rand(4)`, which displays a 4x4 matrix of random values. Below this, the command `matrix2latex3(matrix, 'Aout', 'Mstyle','bmatrix');` is entered, followed by a second prompt line. The window has standard MATLAB window controls (minimize, maximize, close) in the top right corner. At the bottom of the window, there are icons for the MATLAB logo and a green checkmark.

```
>> matrix=rand(4)
matrix =
    0.2769    0.6948    0.4387    0.1869
    0.0462    0.3171    0.3816    0.4898
    0.0971    0.9502    0.7655    0.4456
    0.8235    0.0344    0.7952    0.6463
>> matrix2latex3(matrix, 'Aout', 'Mstyle','bmatrix');
>> |
```

Πάυλος Βραχνής
236010



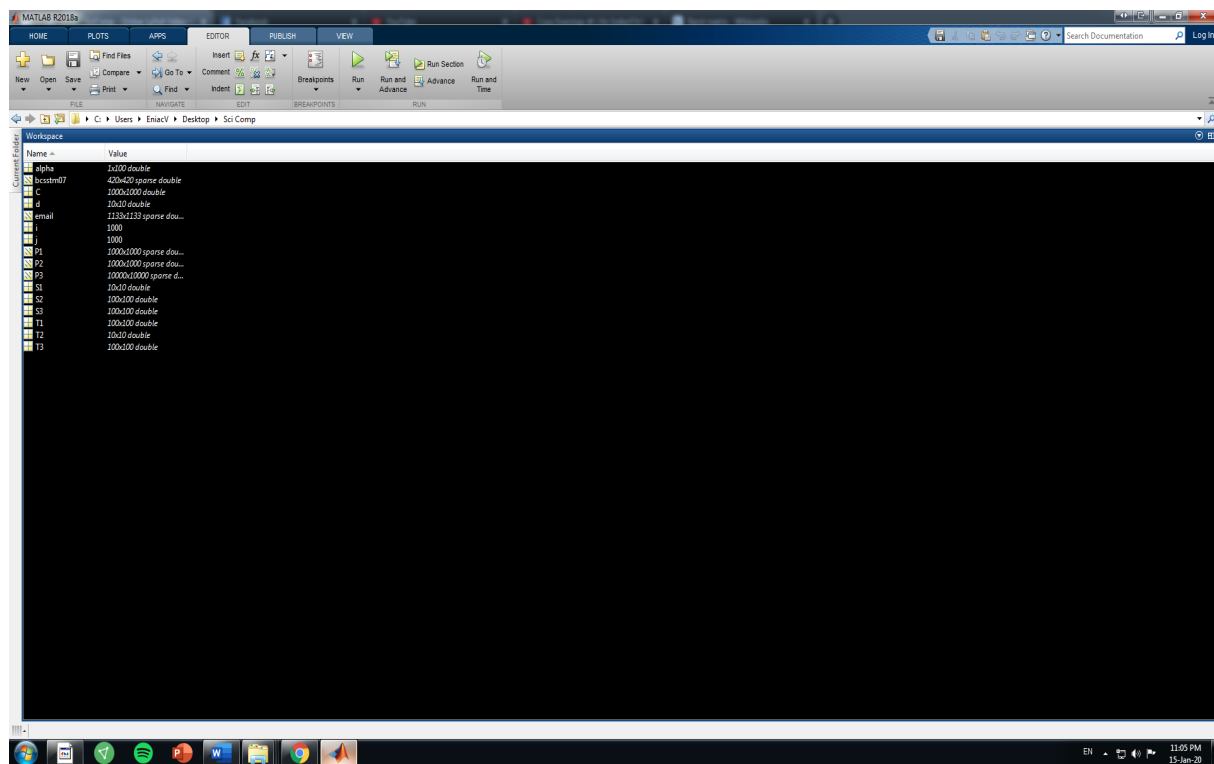


Ερώτημα 3

3.1

Κατασκευάζω τα μητρώα με το αρχείο **create_matrix.m**. Εκτέλεσα την εντολή `create_matrix` και μετά για καθένα από τα μητρώα εκτέλεσα την εντολή `[dflag,discrC,discrR] = dd_check(A)` με όρισμα `A` το όνομα του μητρώου. Τα αποτελέσματα που πήρα δείχνουν αν κάτι ισχύει με το '1' και αν όχι με το '0'. Τις γραμμές τις βρίσκω με την εντολή `size(A,1)` και τις στήλες με την `size(A,2)`. Αν οι γραμμές είναι όσες και οι στήλες τότε το μητρώο είναι συμμετρικό. Το `N` είναι στήλες επί γραμμές. ΔK είναι αν το `dflag=1`. Τον δείκτη κατάστασης τον βρίσκω με την εντολή `condest(M)`. Η αντιστρεψιμότητα ελέγχεται με το αν η ορίζουσα είναι διάφορη του 0. Για το αν είναι μητρώο ζώνης φαίνεται στο αποτέλεσμα της `checkz` και αν είναι, τότε τις υπερδιαγωνίους και υποδιαγωνίους τις βλέπουμε στο `zone(1),zone(2)` αντίστοιχα. Τα αποτελέσματα για τον πίνακα 2 τα παίρνω αλλάζοντας την συνάρτηση για ευκολία σε `[N,dflag,reversable,symmetric,dk,zone,checkz] = dd_check(C.)`. Αλλιώς απλά τυπώνω τα αποτελέσματα. Παραθέτω και μία εκτέλεση της εντολής για ένα εκ των αποτελεσμάτων.

Παραθέτω το workspace μετά την εκτέλεση της εντολής:



Πίνακας 2: Χαρακτηριστικά μητρώων ελέγχου.

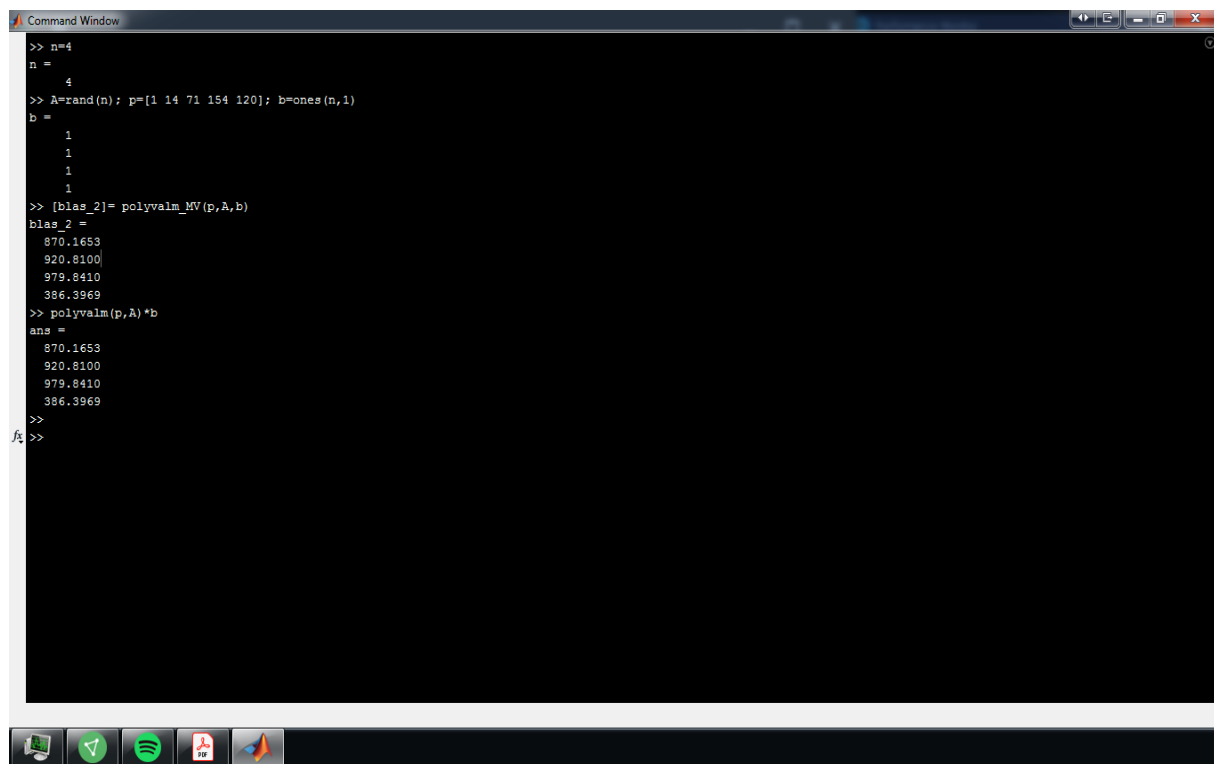
| matrix | N | ΔK | συμμετρικό | ζώνης | αντιστρέψιμο | δ.δ. k1(A) |
|--------------------------------|------|-----|------------|-----------|--------------|------------|
| toeplitz ([2,-1, zeros(1,8)]) | 10 | Ναι | Ναι | (1,1) | Ναι | 60 |
| C1000 | 1000 | Όχι | Ναι | (999,999) | Ναι | 41.2165 |
| P(100,10) | 1000 | Όχι | Ναι | Όχι | Ναι | 139.9998 |
| P(10,100) | 1000 | Όχι | Ναι | Όχι | Ναι | 135.6683 |
| P(100,100) | 1000 | Όχι | Ναι | Όχι | Ναι | 7.0018e+03 |
| bcststm07 | 420 | Όχι | Ναι | Όχι | Ναι | 1.3365e+04 |
| email | 1133 | Όχι | Ναι | Όχι | Όχι | Inf |

3.2

Για να εκμεταλλευτούμε την πράξη BLAS-2 θα πρέπει να πολλαπλασιάσουμε το μητρώο A με το b και όχι να εκτελέσουμε πρώτα την δύναμη. Αρχικά κάνουμε την πράξη $A*b$ και το διάνυσμα που θα πάρουμε ως αποτέλεσμα θα το πολλαπλασιάσουμε πάλι με A όσες φορές είναι η δύναμή του A . Έπειτα θα πολλαπλασιάσουμε το αποτέλεσμα με τον συντελεστή του $p(i)$. Το αποτέλεσμα που θα πάρουμε θα είναι ένα διάνυσμα που θα το προσθέσουμε με τα υπόλοιπα που θα πάρουμε ως αποτέλεσμα. Για να ελέγξουμε την ορθότητα του αποτελέσματος συγκρίνουμε με το αποτέλεσμα της $\text{polyvalm}(p,A)*b$. Η συνάρτηση `polyvalm_MV` κάνει ακριβώς αυτό. Πιο αναλυτικά:

$$1*\zeta^4*b + \dots + 120*b \Rightarrow 1*A^4*b + \dots + 120*b \Rightarrow 1*(A*(A*(A*(A*b)))) + \dots + 120*b$$

Παραθέτω μία εκτέλεση της εντολής και σύγκριση για ορθότητα:



```
>> n=4
n =
     4
>> A=rand(n); p=[1 14 71 164 120]; b=ones(n,1)
b =
     1
     1
     1
     1
>> [blas_2]= polyvalm_MV(p,A,b)
blas_2 =
    870.1653
    920.8100
    979.8410
    386.3969
>> polyvalm(p,A)*b
ans =
    870.1653
    920.8100
    979.8410
    386.3969
>>
>>
```

3.2.1

Στο αρχείο script (έχω βάλει συνάρτηση σε σχόλιο `function [error,timer]=script(A)` για να δίνω εύκολα το όρισμα `script(A)`). Αλλιώς κάθε φορά γράφω `A=` με το μητρώο που θέλουμε.) αρχικά θέτω του συντελεστές του `p` και βρίσκω τις ρίζες του. Έπειτα χρησιμοποιώ το προηγούμενο ερώτημα και θέτω το `b`: `b = polyvalm(MV(p,A,ones(n,1)))` και το `C`: `C= polyvalm(p,A)`. Έπειτα θέτω το `d` ως ένα μητρώο με '1' στη κύρια διαγώνιο. Στη συνέχεια για την `backslash explicit` χρονομετρώ την πράξη `result = C \ b` με χρήση `tic, toc` με 50 επαναλήψεις και διαιρώ με 50 για πιο ορθά αποτελέσματα. Επίσης βρίσκω το `error` με την εντολή `error(1)=norm(result-ones(n,1))` και το χρόνο με την εντολή `timer(1)=temp/50`, όπου `temp=toc`; . Με την ίδια λογική κάνω και τα υπόλοιπα και το μόνο που αλλάζει είναι το πως βρίσκω το `result`.

Serial + backslash: `result=(A-r(length(r)-i+1)*d) \ b`.

Serial + pcg: `C=A-r(length(r)-i+1)*d, result=pcg (C,b,tol,50)`, και `tol=1e-7`

Serial + pcg με προρύθμιση μπλοκ Jacobi: `C=A-r(length(r)-i+1)*d, result=pcg(C,b,tol,50,diag(diag(C)))`, με `diag(diag(C))` μητρώο για προρύθμιση Jacobi και `tol=1e-7`

Για να βρω τα αποτελέσματα κρατάω όλους τους χρόνους στην `timer` και όλα τα `errors` στην `error`. Και κάθε φορά καλώ την `[error,timer]=script(A)` με `A` το μητρώο που θέλουμε ως είσοδο.

*Αν δεν θέλουμε την `script` ως `function` τότε απλά θέτουμε `A=` μητρώο που θέλουμε και αφαιρούμε την εντολή `function [error,timer]=script(A)`.

Παραθέτω μία εκτέλεση της εντολής για $A=C$:

```

Command Window
pcg converged at iteration 6 to a solution with relative residual 2.2e-08.
pcg converged at iteration 7 to a solution with relative residual 1.4e-08.
pcg converged at iteration 6 to a solution with relative residual 6.5e-08.
pcg converged at iteration 6 to a solution with relative residual 3.6e-08.
pcg converged at iteration 6 to a solution with relative residual 2.2e-08.
pcg converged at iteration 7 to a solution with relative residual 1.4e-08.
pcg converged at iteration 6 to a solution with relative residual 6.5e-08.
pcg converged at iteration 6 to a solution with relative residual 3.6e-08.
pcg converged at iteration 6 to a solution with relative residual 2.2e-08.
pcg converged at iteration 7 to a solution with relative residual 1.4e-08.
pcg converged at iteration 6 to a solution with relative residual 6.5e-08.
pcg converged at iteration 6 to a solution with relative residual 3.6e-08.
pcg converged at iteration 6 to a solution with relative residual 2.2e-08.
pcg converged at iteration 7 to a solution with relative residual 1.4e-08.
pcg converged at iteration 6 to a solution with relative residual 6.5e-08.
pcg converged at iteration 6 to a solution with relative residual 3.6e-08.
pcg converged at iteration 6 to a solution with relative residual 2.2e-08.
Elapsed time is 2.458414 seconds.
>> timer
timer =
    0.0135    0.0455    0.0323    0.0492
>> error
error =
    1.0e+06 *
    0.0000    1.0142    1.0142    1.0142
>> error(1)
ans =
    9.5053e-14
>> error(2)
ans =
    1.0142e+06
>> error(3)
ans =
    1.0142e+06
>> error(4)
ans =
    1.0142e+06
>>

```

Πίνακας 3: Χρόνοι εκτέλεσης.

| RUNTIMES (sec) | C_{1000} | $P_{(10,100)}$ | $P_{(100,10)}$ | $P_{(100,100)}$ | bcsstm07 |
|-------------------------------|------------|----------------|----------------|-----------------|----------|
| explicit | 0.0135 | 0.00995 | 0.0099 | 2.8965 | 0.0020 |
| serial+backslash | 0.0455 | 0.0551 | 0.0552 | 13.5387 | 0.0093 |
| serial+Cholesky | X | | | | |
| serial+PCG | X | 0.0617 | 0.0566 | 6.763 | 0.0166 |
| serial+PCG+prec(blockJ) | X | 0.1611 | 0.1505 | 15.4864 | 0.0271 |
| parallel+backslash (προ-2014) | | | | | |

Πίνακας 4: Νόρμες-1 σφαλμάτων $\|e - x\|_2$.

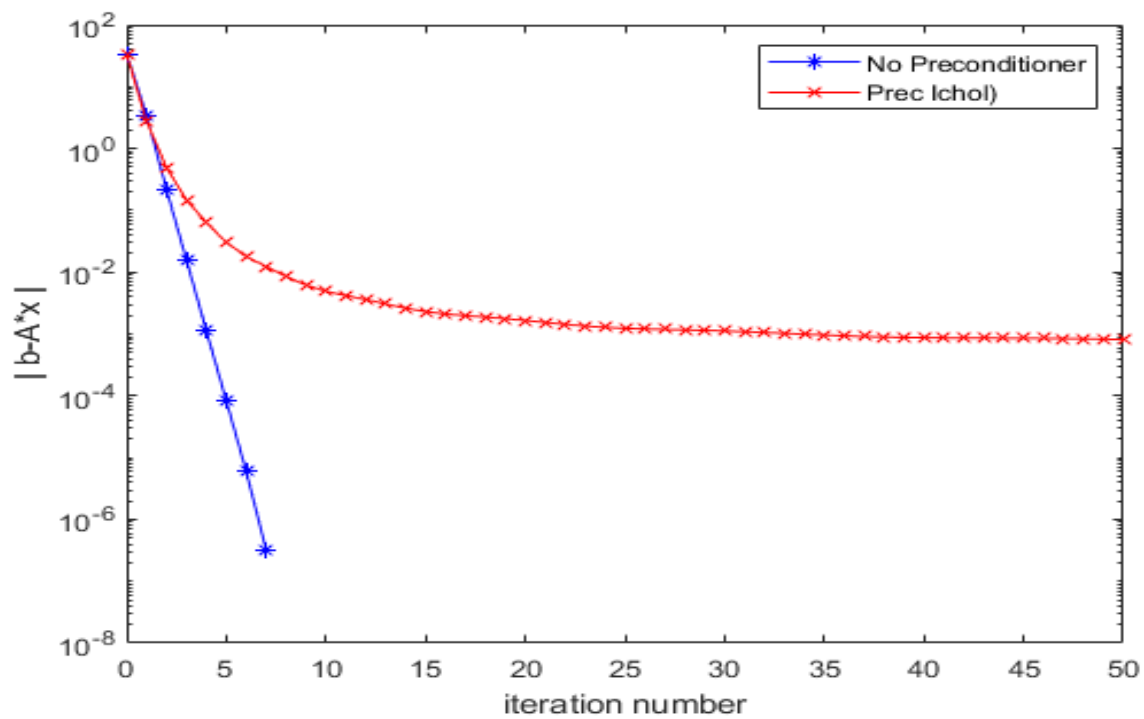
| ERRORS ($\ \cdot\ _2$) | C ₁₀₀₀ | P _(10,100) | P _(100,10) | P _(100,100) | bcsstm07 |
|-------------------------------|-------------------|-----------------------|-----------------------|------------------------|------------|
| explicit | 9.5053e-14 | 9.6778e-11 | 1.0926e-10 | 2.0405e-09 | 8.7830e-06 |
| serial+backslash | 1.0142e+06 | 2.7064e+05 | 4.4658e+05 | 4.8594e+05 | 2.3145e+09 |
| serial+Cholesky | X | | | | |
| serial+PCG | X | 2.7064e+05 | 4.4658e+05 | 4.8594e+05 | 2.3145e+09 |
| serial+PCG+prec(blockJ) | X | 2.7064e+05 | 4.4658e+05 | 4.8594e+05 | 2.3145e+09 |
| parallel+backslash (προ-2014) | | | | | |

Ερώτημα 4

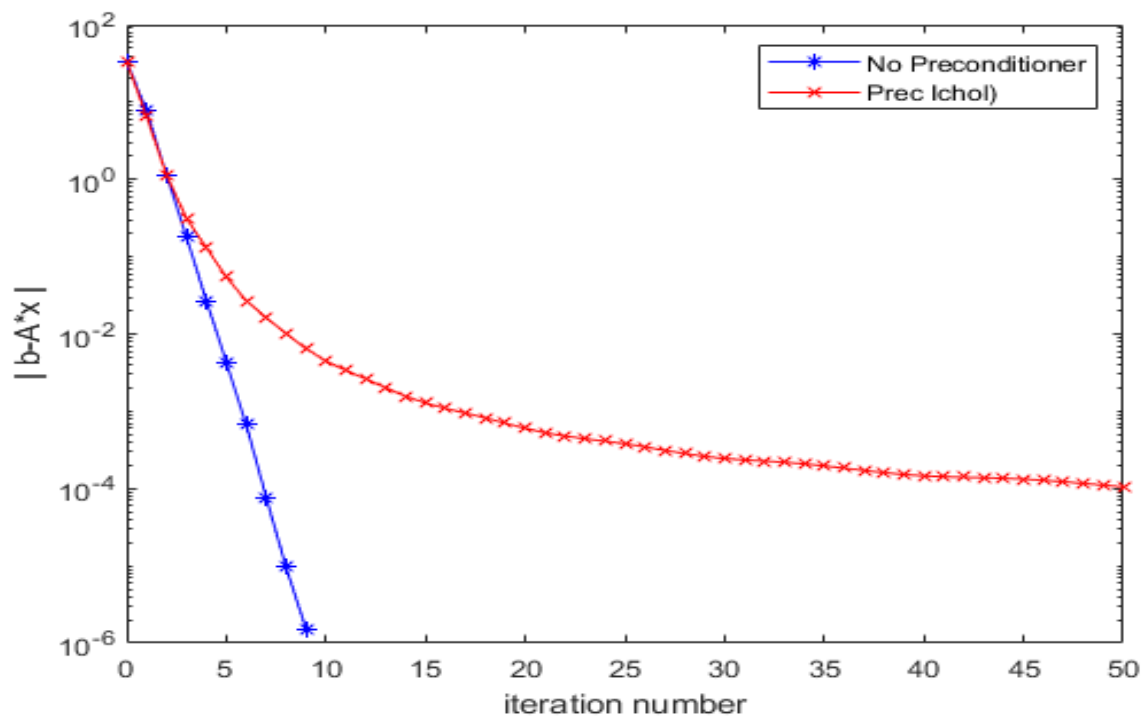
Συνάρτηση multiKatz: Σε αυτή τη συνάρτηση αρχικά κοιτάω το mth. Αν είναι 'direct', τότε για κάθε a εκτελώ την πράξη $(I - \alpha(i) * A)$ και την αποθηκεύω στο temp, στη συνέχεια εξετάζω αν το temp έχει ορίζουσα διάφορη του μηδέν. Αν έχει τότε εξετάζω αν το αποτέλεσμα της πράξης temp e που έχει όλα τα στοιχεία θετικά. Αν ναι τότε η τιμή του a είναι έγκυρη και κρατάω το a στο values το πόσες τιμές έχω στο count και το αποτέλεσμα προσθέτεται σα στήλη στο result. Αν το mth είναι 'pcg' τότε εκτελώ τα παραπάνω βήματα, αλλά αντί για την πράξη temp e κάνω την $[temp, fl0, rr0, it0, rv0] = pcg(sparse(temp), e, pcg_parms1, pcg_parms2)$. Για pcg με προρύθμιση ichol πρέπει να δώσω παραπάνω από δύο παραμέτρους, δηλ πρέπει να δώσω και 1 ή 2 μητρώα για προρύθμιση. Τέλος τυπώνω το X αλλά επιπλέον τυπώνω και τα αποτελέσματα που ζητούνται στον πίνακα και έχω σε σχόλιο την $[X, time, values, max_A, count, y, it0] = multiKatz(A, alpha, mth, pcg_parms)$ για να παίρνω τα ζητούμενα αποτελέσματα, τα οποία εξηγώ παρακάτω.

Διάγραμματα

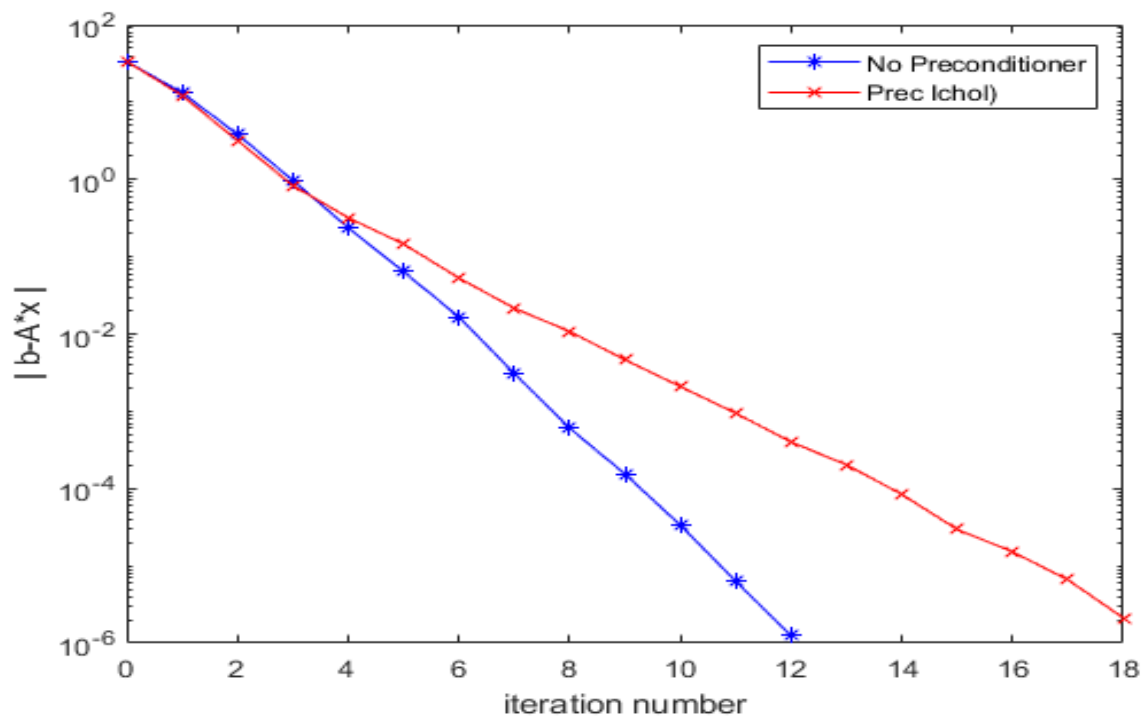
$a=0.01$



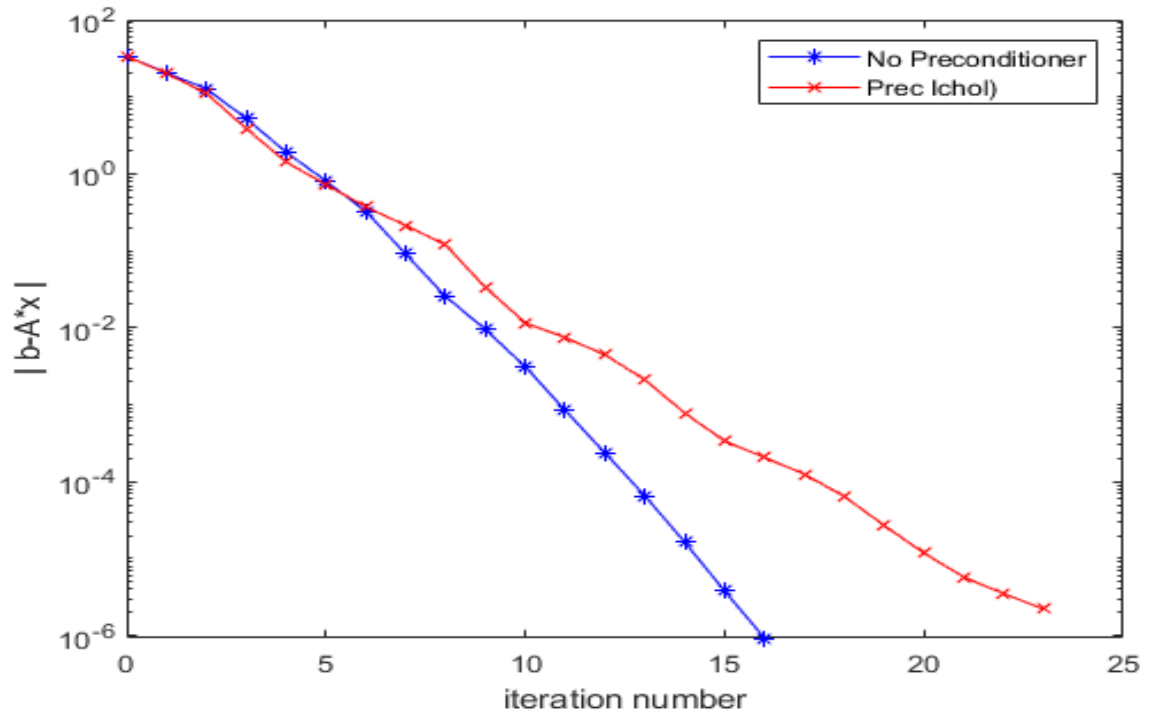
$a=0.02$



a=0.03



$a=0.04$



Τα διαγράμματα βρέθηκαν εκτελώντας την εντολή
`[values,values2,max_A,max_A2]=multiKatz(email,alpha,"pcg",10^(-7),50,L,L)`
για να βρω τα a με και χωρίς προρυθμιστή αντιστοιχα και μετά με την εντολή
`multiKatz(email,a,"pcg",10^(-7),50,L,L)` όπου a η έγκυρη τιμή του κάθε a και
`temp2=(I-alpha(i)*A); L=ichol(sparse(temp2))` με I μητρώο με '1' στην κύρια
διαγώνιο και `length==length(A)`. Ακόμα πρόσθεσα τις εξής εντολές στον
κώδικα: (τις έχω σε σχόλιο)

```
figure;
semilogy(0:it0,rv0,'-b');
hold on;
semilogy(0:it2,rv2,'-r');
legend('No Preconditioner','Prec Ichol');
xlabel('iteration number');
ylabel('||b - A * x||');
hold off;
```

Τέλος άλλαξα τη δομή του κώδικα αφού έπρεπε να τρέξουν και οι δύο `pcg` μαζί.

(Για να τρέξει και να προβάλλει τα διαγράμματα πρέπει να γίνουν κάποιες τροποποιήσεις γιατί αλλιώς δεν θα δούλευε σωστά και όπως ζητήσατε η συνάρτηση.

Ουσιαστικά έβγαλα το `if` ώστε να τρέχουν και οι δύο αν δοθούν περισσότερες

από 2 παράμετροι, κράτησα τα δεδομένα του pcg + ichol ως έχουν και πρόσθεσα τον παρπάνω κώδικα.)

Πίνακας 5

| 1-alpha(1)*email | a | runtime(sec) | επαναλήψεις | $\ b - A * x\ _2$ | top-5 (hi to lo) |
|------------------------|--------|--------------|-------------|-------------------|----------------------------|
| explicit | 0.0100 | 0.0579 | X | 537.7980 | (105,333,42,16,23) |
| | 0.0400 | 0.0781 | X | 2.4501e+03 | (105,16,42,196,333) |
| serial+PCG | 0.0100 | 0.0506 | 7 | 537.7980 | (105,333,42,16,23) |
| | 0.0500 | 0.0620 | 1 | 846.0450 | (1133,1132,1131,1130,1129) |
| serial+PCG+prec(ichol) | 0.01 | 0.0530 | 50 | 537.7980 | (105, 333, 42, 16, 23) |
| | 0.04 | 0.0494 | 23 | 2.4501e+03 | (105, 16, 42, 196, 333) |

Η μέγιστη τιμή του α βρέθηκε από την εντολή values(count) καθώς το values κρατάει όλες τις τιμές που έχει νόημα ο υπολογισμός και το count μετράει το πόσες είναι, άρα η τιμή values(count) θα είναι η μεγαλύτερη τιμή του α . Το runtime βρέθηκε με εντολές tic, toc στην αρχή και στο τέλος του κώδικα και αποθηκεύεται στην μεταβλητή times. Οι επαναλήψεις από την μεταβλητή it0 από την pcg. Η νόρμα με την εντολή norm((b-A*result),2) και το top-5 (hi to lo) με την εντολή [x,y]=sort(result(:1129:1133,1)) και κρατάμε τα 5 στην y με την εντολή y=y(1129:1133).

Παρακάτω δείχνω τις εκτελέσεις για όρισμα alpha για να βρω όλες τις έγκυρες τιμές και μία εκτέλεση για την τιμή 0.01 με pcg.

Πάυλος Βραχνής
236010

Παραθέτω μία εκτέλεση της εντολής για `[X,temp,values,max_A,count,y]=multiKatz(email,alpha,"direct")`:

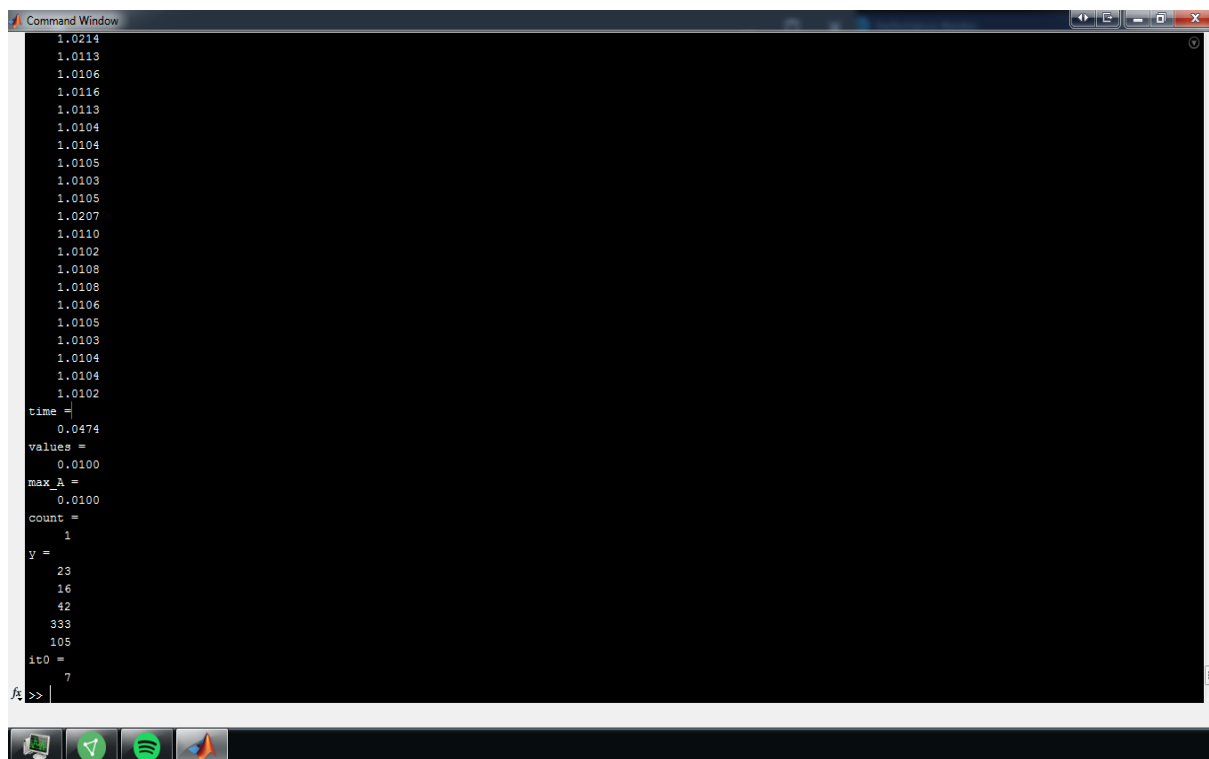
```
Command Window
1.0114 1.0266 1.0495 1.1011
1.0660 1.1483 1.2600 1.4532
1.0214 1.0470 1.0817 1.1530
1.0113 1.0263 1.0489 1.0985
1.0106 1.0232 1.0402 1.0735
1.0116 1.0287 1.0591 1.1460
1.0113 1.0269 1.0522 1.1173
1.0104 1.0217 1.0352 1.0562
1.0104 1.0217 1.0353 1.0568
1.0105 1.0225 1.0382 1.0683
1.0103 1.0215 1.0342 1.0513
1.0105 1.0223 1.0372 1.0635
1.0207 1.0433 1.0684 1.0998
1.0110 1.0251 1.0458 1.0910
1.0102 1.0210 1.0326 1.0464
1.0108 1.0238 1.0410 1.0713
1.0108 1.0236 1.0400 1.0672
1.0106 1.0227 1.0374 1.0588
1.0105 1.0224 1.0363 1.0551
1.0103 1.0217 1.0348 1.0539
1.0104 1.0219 1.0347 1.0501
1.0104 1.0218 1.0346 1.0503
1.0102 1.0209 1.0322 1.0447
temp =
4.8601
values =
0.0100 0.0200 0.0300 0.0400
max_A =
0.0400
count =
4
y =
23
16
42
333
105
fx >>
```

Παραθέτω μία εκτέλεση της εντολής για $[X, \text{time}, \text{values}, \text{max_A}, \text{count}, y] = \text{multiKatz}(\text{email}, \alpha, \text{"pcg"}, 10, \hat{(-7)}, 50)$

```
Command Window
1.0114 1.0266 1.0495 1.1011 1.9272
1.0660 1.1483 1.2600 1.4532 1.9272
1.0214 1.0470 1.0817 1.1530 1.9272
1.0113 1.0263 1.0489 1.0985 1.9272
1.0106 1.0232 1.0402 1.0735 1.9272
1.0116 1.0287 1.0591 1.1460 1.9272
1.0113 1.0269 1.0522 1.1173 1.9272
1.0104 1.0217 1.0352 1.0562 1.9272
1.0104 1.0217 1.0353 1.0568 1.9272
1.0105 1.0225 1.0382 1.0683 1.9272
1.0103 1.0215 1.0342 1.0513 1.9272
1.0105 1.0223 1.0372 1.0635 1.9272
1.0207 1.0433 1.0684 1.0998 1.9272
1.0110 1.0251 1.0458 1.0910 1.9272
1.0102 1.0210 1.0326 1.0464 1.9272
1.0108 1.0238 1.0410 1.0713 1.9272
1.0108 1.0236 1.0400 1.0672 1.9272
1.0106 1.0227 1.0374 1.0588 1.9272
1.0105 1.0224 1.0363 1.0551 1.9272
1.0103 1.0217 1.0348 1.0539 1.9272
1.0104 1.0219 1.0347 1.0501 1.9272
1.0104 1.0218 1.0346 1.0503 1.9272
1.0102 1.0209 1.0322 1.0447 1.9272
time =
1.9742
values =
0.0100 0.0200 0.0300 0.0400 0.0500
max_A =
0.0500
count =
5
y =
23
16
42
333
105
f3 >>
```

Πάυλος Βραχνής
236010

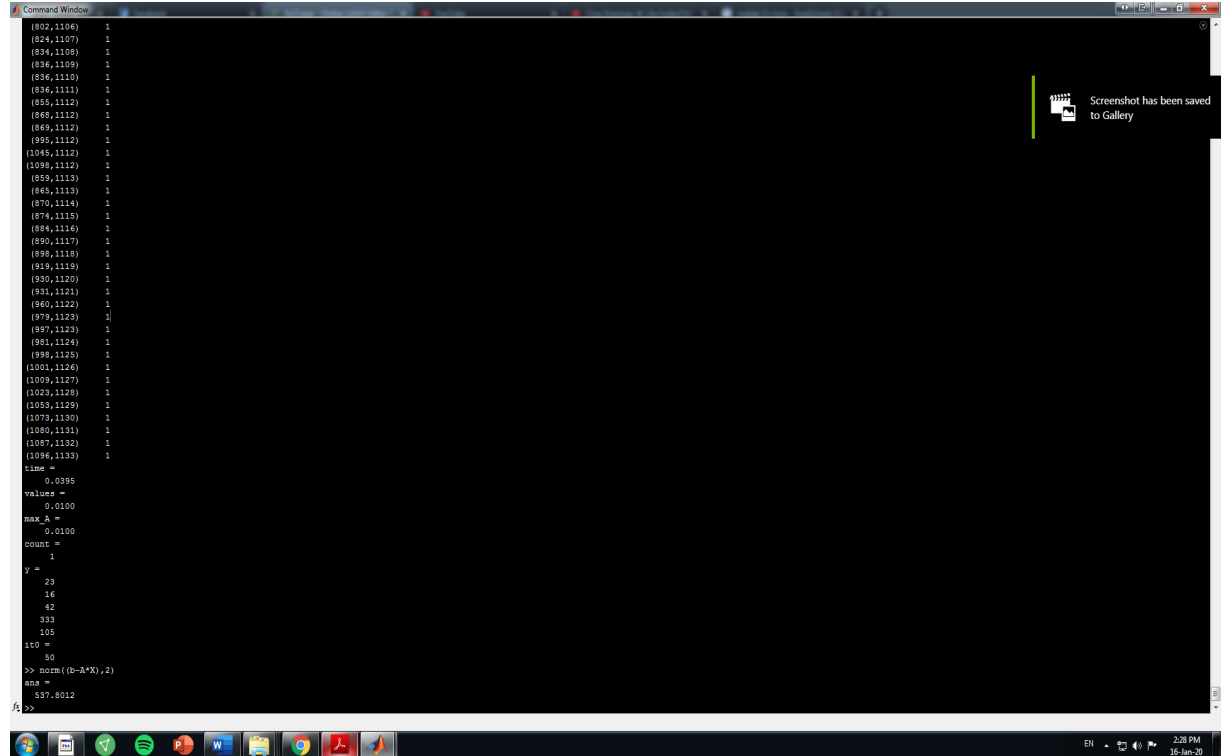
Παραθέτω μία εκτέλεση της εντολής για $[X, \text{time}, \text{values}, \text{max_}$
 $A, \text{count}, y, \text{it0}] = \text{multiKatz}(\text{email}, 0.01, \text{'pcg'}, 10, \hat{-7}, 50)$



```
Command Window
1.0214
1.0113
1.0106
1.0116
1.0113
1.0104
1.0104
1.0104
1.0105
1.0103
1.0105
1.0207
1.0110
1.0102
1.0108
1.0108
1.0106
1.0105
1.0103
1.0104
1.0104
1.0102
time =
0.0474
values =
0.0100
max_A =
0.0100
count =
1
y =
23
16
42
333
105
it0 =
7
>>
```

Πάυλος Βραχνής
236010

Παραθέτω και μία εκτέλεση της εντολής με προρύθμιση ichol για $a=0.01$ με
εντολές πριν την εκτέλεση: $r=length(email);$
 $I=eye(r);$
 $L=ichol(sparse(I-0.01*email));$



The screenshot shows a Windows Command Window with the following output from MATLAB:

```
(802,1106) 1
(824,1107) 1
(834,1108) 1
(836,1109) 1
(836,1110) 1
(836,1111) 1
(855,1112) 1
(868,1112) 1
(869,1112) 1
(895,1112) 1
(1045,1112) 1
(1080,1112) 1
(829,1113) 1
(845,1113) 1
(870,1114) 1
(874,1115) 1
(884,1116) 1
(890,1117) 1
(898,1118) 1
(919,1119) 1
(930,1120) 1
(931,1121) 1
(960,1122) 1
(979,1123) 1
(997,1123) 1
(981,1124) 1
(988,1125) 1
(1001,1126) 1
(1009,1127) 1
(1023,1128) 1
(1053,1129) 1
(1079,1130) 1
(1080,1131) 1
(1087,1132) 1
(1096,1133) 1
time =
    0.9395
valued =
    0.0100
max_A =
    0.0100
count =
     1
y =
    23
    16
    42
    333
    109
it0 =
    50
>> norm((D-A*X),2)
ans =
    537.8012
>>
```

A notification on the right side of the window states: "Screenshot has been saved to Gallery". The Windows taskbar at the bottom shows various application icons and the system clock indicating 2:38 PM on 10-Apr-20.