# Επιστημονικός Υπολογισμός I

# 1η εργαστηριακή άσκηση

## Ερώτημα 1 - Εισαγωγικά

(i) Υπολογιστικό σύστημα: Σταθερός υπολογιστής

Τύπος και συχνότητα επεξεργαστή: AMD FX-8350 4 GHz

Αριθμός και μέγεθος κρυφών επιπέδων: 3 Μεγέθη

Level 1 cache: 4 x 64 KB instruction caches | Level 2 cache: 4 x 2 MB shared | Level 3 cache: 8 MB shared cache.

Κύρια μνήμη DDR3 8GB

(iii) Microsoft 10 Pro N 64-bit (10.0, Build 10240)

Matlab R2015a Version : 8.5 Build 197 613

(iv) Matlab Benchmark

================

Number of times each test is run___________________________: 15

I. Matrix calculation

---------------------

Creation, transp., deformation of a 1200x1200 matrix (sec): 0.060509

1250x1250 normal distributed random matrix ^1000____ (sec): 0.059076

Sorting of 1,100,000 random values__________________ (sec): 0.039041

550x550 cross-product matrix (b = a' * a)___________ (sec): 0.0090426

Linear regression over a 700x700 matrix (c = a \ b') (sec): 0.015625

--------------------------------------------

Trimmed mean (2 extremes eliminated): 0.037914

II. Matrix functions

--------------------

FFT over 900,000 random values_______________________ (sec): 0.043481

Eigenvalues of a 220x220 random matrix______________ (sec): 0.057683

Determinant of a 750x750 random matrix______________ (sec): 0.011918

Cholesky decomposition of a 1000x1000 matrix________ (sec): 0.013355

Inverse of a 500x500 random matrix__________________ (sec): 0.01735

--------------------------------------------

Trimmed mean (2 extremes eliminated): 0.024729

III. Programmation

------------------

225,000 Fibonacci numbers calculation (vector calc)_ (sec): 0.018692

Creation of a 1500x1500 Hilbert matrix (matrix calc) (sec): 0.055888

Grand common divisors of 35,000 pairs (recursion)___ (sec): 0.0118

Creation of a 220x220 Toeplitz matrix (loops)_______ (sec): 0.00038012

Escoufier's method on a 22x22 matrix (mixed)________ (sec): 0.11293

-------------------------------------------

Trimmed mean (2 extremes eliminated): 0.028793

Total time for all 15 tests_________________________ (sec): 0.52678

Overall mean (sum of I, II and III trimmed means/3)_ (sec): 0.030479

--- End of test ---

## Ερώτημα 2 – Χρονομέτρηση Συναρτήσεων

(i) Οι βασικές πράξεις γραμμικής άλγεβρας που υλοποιούνται με τις παρακάτω συναρτήσεις είναι οι εξής:

Lu= Εκφράζει το μητρώο Α ως γινόμενο δύο στοιχειωδών τριγωνικών μητρώων, ένα μεταθετικό κάτω τριγωνικό και ένα άνω τριγωνικό μητρώο.

Qr= Είναι η ορθοφώνια τριγωνική διάσπαση ενός μητρώου.

Svd= Αναλύει τον πίνακα σε ιδιάζουσες τιμές.

Det= Επιστρέφει την ορίζουσα του πίνακα.

Rank= Επιστρέφει την τάξη του μητρώου.

Polyval= Επιστρέφει την τιμή ενός πολυώνυμου βαθμού n συναρτήσει του x. Οι συντελεστές είναι διανύσματα ενώ το x μπορεί να είναι και διάνυσμα ή μητρώο.

Ενδογενείς συναρτήσεις είναι οι lu,qr,svd,det ενώ οι rank και polyval είναι m-functions.

Η συνάρτηση rank χρησιμοποίει τις :

max,size,eps,nargin,sum και svd ενδογενείς.

Η polyval χρησιμοποιεί τις:

length,nargin,nargout,isfinite,isscalar,all,filter,size,zeros,isempty,isstruct,ones,class,sum,sqrt ,inf και reshape.

(ii) Για τυχαίο τετραγωνικό μητρώο Α n by n και διάνυσμα b n by 1 εκτελέστηκαν οι πράξεις [L.U.P]=lu(A) και c=A*b με τους παρακάτω τρόπους.

(α) Script(E22a)

```
for i=[7:12]

    n= 2.^[i];

    u= rand(n,1);
    v= rand(n,1);
    b= rand(n,1);

    f=@() rank2_power(u,v,b);
    T(1,i-6)= timeit(f);
    flops= 18*n^3 -4*n^2-n;
    F(1,i-6)= flops/T(1,i-6)*10^9;

    f=@() my_rank2_power(u,v,b);
    T(2,i-6)= timeit(f);
    flops= 23*n^2 -10*n;
    F(2,i-6)= flops/T(2,i-6)*10^9;

end

n=2.^[7:12];

figure %Xronoi
plot(n,T(1,:),'bx--')
hold on
plot(n,T(2,:),'k*-')
hold off
legend('rank2power x', 'myrank2power *')
title('Xronoi ekteleshs twn dyo function')
xlabel('matrix size')
```

```
ylabel('time (s)')

figure %Epidosi
plot(n,F(1,:),'bx--')
hold on
plot(n,F(2,:),'k*-')
hold off
legend('rank2power x','myrank2power *')
title('Epidosi ekteleshs twn dyo function')
xlabel('matrix size')
ylabel('Glops/s')
```

Με την χρήση των συναρτήσεων tic,toc και εκτελώντας κάθε πράξη μόνο μια φορά. Φτιάξαμε ένα τυχαίο μητρώο A και ένα διάνυσμα b με την βοήθεια της rand. Σε ένα μητρώο με όνομα T1 και σε ένα άλλο με όνομα F1 αποθηκεύτηκαν οι τιμές των χρόνων και των flops αντίστοιχα για τις μετρήσεις της lu(A) και στους T2 και F2 τα αντίστοιχα της πράξης c=A*b.

Τα μητρώα με τους χρόνους και με τα flops αντίστοιχα είναι τα εξής:

Lu(A)

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 5.5909e-04 | 0.0026 | 0.0088 | 0.0415 | 0.1606 | 1.0384 |

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 2.5007e+18 | 4.2293e+18 | 1.0112e+19 | 1.7252e+19 | 3.5653e+19 | 4.4118e+19 |

c=A*b

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 3.1089e-05 | 5.1220e-05 | 2.9254e-04 | 0.0018 | 0.0056 | 0.0306 |

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1.0499e+18 | 2.5540e+18 | 1.7904e+18 | 1.1487e+18 | 1.4907e+18 | 1.0961e+18 |

(β) (E22b)

```
T3= zeros (6,10);
T4= zeros (6,10);
F3= zeros(6,10);
F4= zeros(6,10);
S= zeros (1,6);
for i= 7:12
   n=2.^i;
   A= rand (n,n);
   B= rand (n,1);
   S(1,i-6)= n;
   for j= 1:10 %deka epanalipsis
      tic
      [L, U, P]=lu (A);
      T3(i-6,j)=toc; %se kathe stili apothikeuetai h timh ths epanalipsis j
      flops= (2/3)*n^3;
      F3(i-6,j)= flops/T3(i-6,j)*10^9;

      tic
      c= A*B;
      T4(i-6,j)=toc;
      flops= 2*n^2-n;
      F4(i-6,j)= flops/T4(i-6,j)*10^9;

   end
end

T3= (mean(T3,2))'; %upologismos mesou orou kai anastrofh mhtrwwn
T4= (mean(T4,2))';
F3= (mean(F3,2))';
```

```
F4= (mean(F4,2))';

figure
plot(S,T3,'b*--')
hold on
plot(S,T4,'rx-')
hold off
legend('lu(A)','c=A*b')
title('Xronos 10 ektelesewn')
xlabel('matrix size')
ylabel('time(s)')

figure
plot(S,F3,'b*--')
hold on
plot(S,F4,'rx-')
hold off
legend('lu(A) *', 'c=A*b x')
title('Epidosi 10 ektelesewn')
xlabel('matrix size')
ylabel('Gflops/s')
```

Οι πράξεις εκτελέστηκαν περισσότερες φορές με την χρήση της tic,toc. Για να γίνει αυτό χρησιμοποιήθηκε μια εμφωλευμένη ‘for.. end’ έτσι ώστε να δέχεται τα μητρώα και να τα χρονομετρεί 10 φορές. Στο τέλος βρίσκουμε τον μέσο όρο των μετρήσεων και τον αποθηκεύουμε μέσα στα βοηθητικά μας μητρώα.

Τα μητρώα με τους χρόνους και με τα flops αντίστοιχα είναι τα εξής:

Lu(A)

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0.0032 | 0.0016 | 0.0089 | 0.0312 | 0.1739 | 1.0583 |

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 2.8695e+18 | 6.9895e+18 | 1.0146e+19 | 2.2938e+19 | 3.3201e+19 | 4.3311e+19 |

c=A*b

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1.7838e-05 | 5.2927e-05 | 3.4514e-04 | 0.0020 | 0.0062 | 0.0337 |

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 2.3026e+18 | 2.6014e+18 | 1.5566e+18 | 1.0282e+18 | 1.3446e+18 | 9.9787e+17 |

Για μεγαλύτερη ακρίβεια στις μετρήσεις είναι προτιμότερο να επαναλαμβάνουμε περισσότερες από μια φορές την εκτέλεση καθώς την πρώτη φορά εμπεριέχεται ο χρόνος που χρειάζεται για να γίνει το LOAD του μητρώου. Έτσι με περισσότερες επαναλήψεις ο χρόνος της LOAD “εξαφανίζεται” όσο το δυνατόν.

(γ) Script(E22g)

```
T5= zeros (1,6);

T6= zeros (1,6);

F5= zeros (1,6);

F6= zeros (1,6);

S= zeros (1,6);

for i= 7:12

   n=2.^i;

   A= rand (n,n);
```

```
    B= rand (n,1);

    S(1,i-6)= n;


    f1=@()lu (A); %handler ths function

    T5(1,i-6)=timeit(f1); %apothikeusi xronou me xrhsh timeit

    flops= (2/3)*n^3;

    F5(1,i-6)= flops/T5(1,i-6)*10^9;


    f2=@() (A*B);

    T6(1,i-6)=timeit(f2);

    flops= 2*n^2-n;

    F6(1,i-6)=flops/T6(1,i-6)*10^9;


end


figure

plot(S,T5,'b*--')

hold on

plot(S,T6,'rx-')

hold off

legend('lu(A) *','c=A*b x')

title('Xronos me timeit')
```

```
xlabel('matrix size')

ylabel('time(s)')


figure

plot(S,F5,'b*--')

hold on

plot(S,F6,'rx-')

hold off

legend('lu(A) *','c=A*b x')

title('Epidosi me timeit')

xlabel('matrix size')

ylabel('Gflops/s')
```

Σε αυτό το υποερώτημα, χρονομετρήσαμε τις πράξεις με την χρήση της timeit. Η timeit επαναλαμβάνει περισσότερες από μια φορές τις μετρήσεις οπότε δεν χρειάζεται κάποια εμφωλευμένη επανάληψη για την ακρίβειά της. Επίσης σταματάει να κάνει επαναλήψεις όταν αυτή θεωρεί οτι έχουν σταθεροποιηθεί τα αποτελέσματα. Χρησιμοποιήσαμε τον function handler f=@()lu(A)και f=@()(A*B) για τις πράξεις μας αντίστοιχα.

Τα μητρώα με τους χρόνους και με τα flops αντίστοιχα είναι τα εξής:

Lu(A)

Χρόνοι:

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 3.3433e-04 | 0.0014 | 0.0064 | 0.0231 | 0.1446 | 1.0128 |

Flops:

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 4.1818e+18 | 8.2430e+18 | 1.3992e+19 | 3.0966e+19 | 3.9595e+19 | 4.5235e+19 |

c=A*b

Χρόνοι:

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 9.9838e-06 | 3.4555e-05 | 2.2935e-04 | 9.2470e-04 | 0.0051 | 0.0276 |

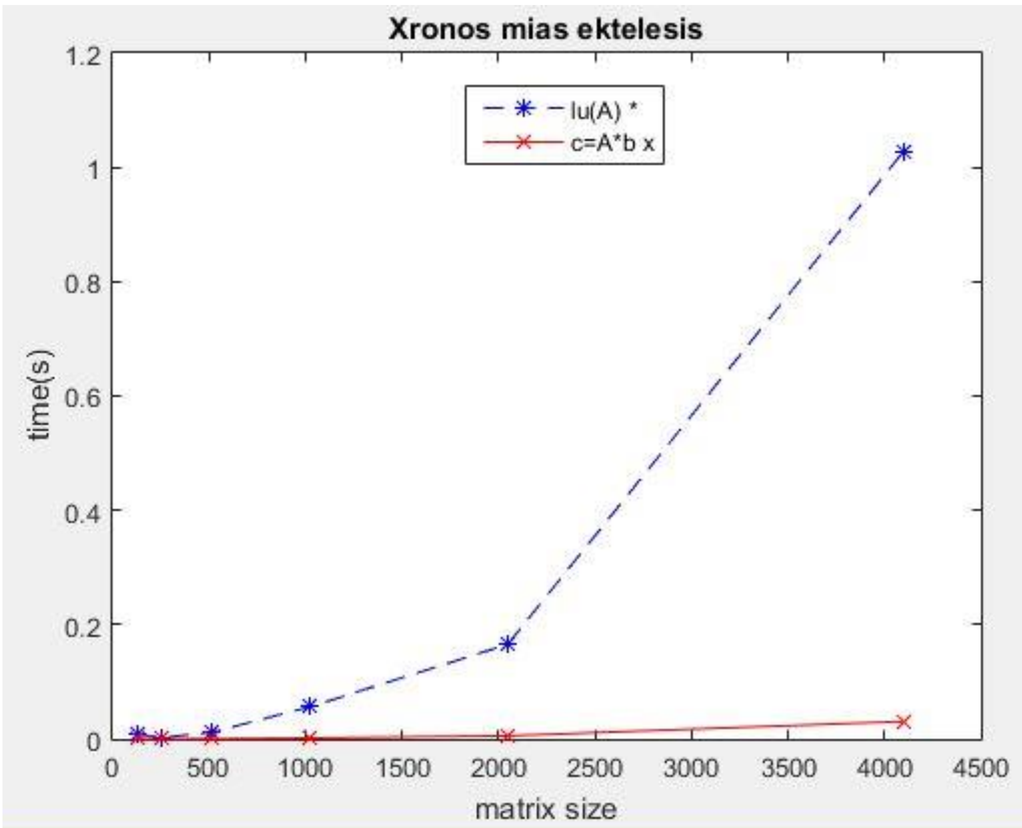
Flops:

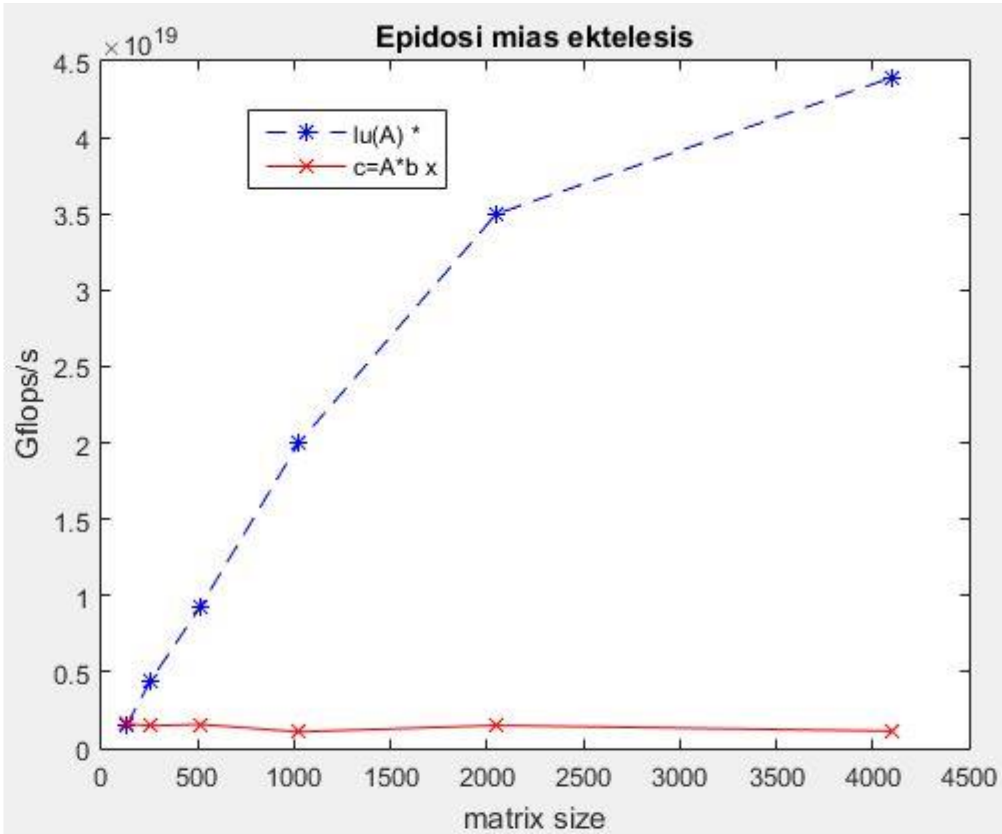
| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 3.2693e+18 | 3.7858e+18 | 2.2837e+18 | 2.2668e+18 | 1.6446e+18 | 1.2142e+18 |

Όπως παρατηρούμε τα αποτελέσματα της timeit διαφέρουν από αυτά της tic,toc με 10 επαναλήψεις. Αυτό μπορεί να συμβαίνει επειδή οι 10 επαναλήψεις μπορεί να μην ήταν αρκετές για να είναι ακριβείς οι μετρήσεις μας.

(iii) Εδώ θα παρουσιαστούν οι γραφικές παραστάσεις για τον κάθε τρόπο μέτρησης των πράξεών μας. Τα σημεία του κώδικα που δημιούγησαν τα figures έχουν δωθεί στα προηγούμενα ερωτήματα

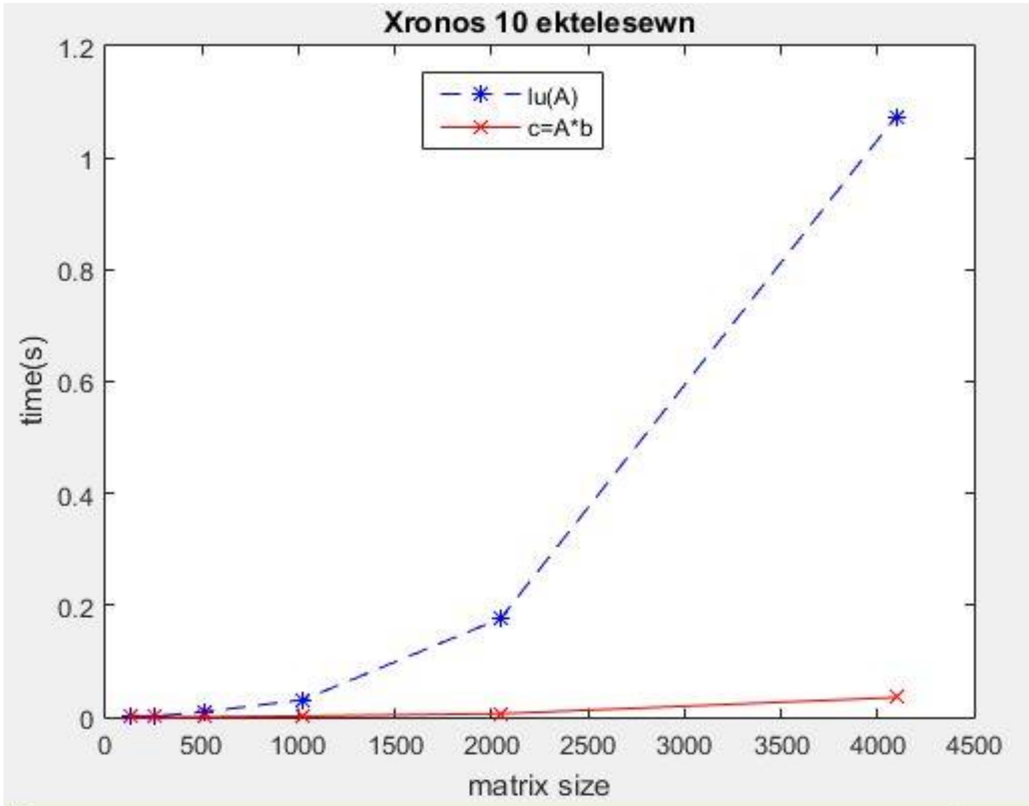
Αρχικά για το (α')

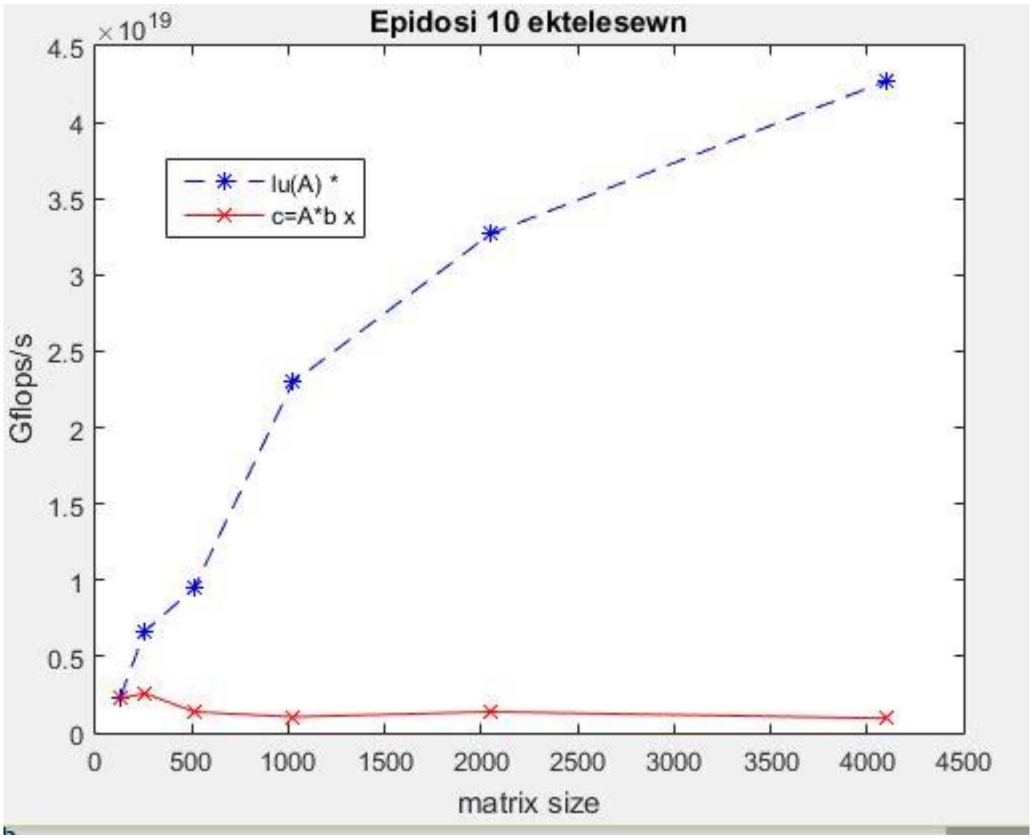




Epidosi mias ektelesis
$\times 10^{19}$
Gflops/s
matrix size
lu(A) *
c=A*b x
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
0
500
1000
1500
2000
2500
3000
3500
4000
4500

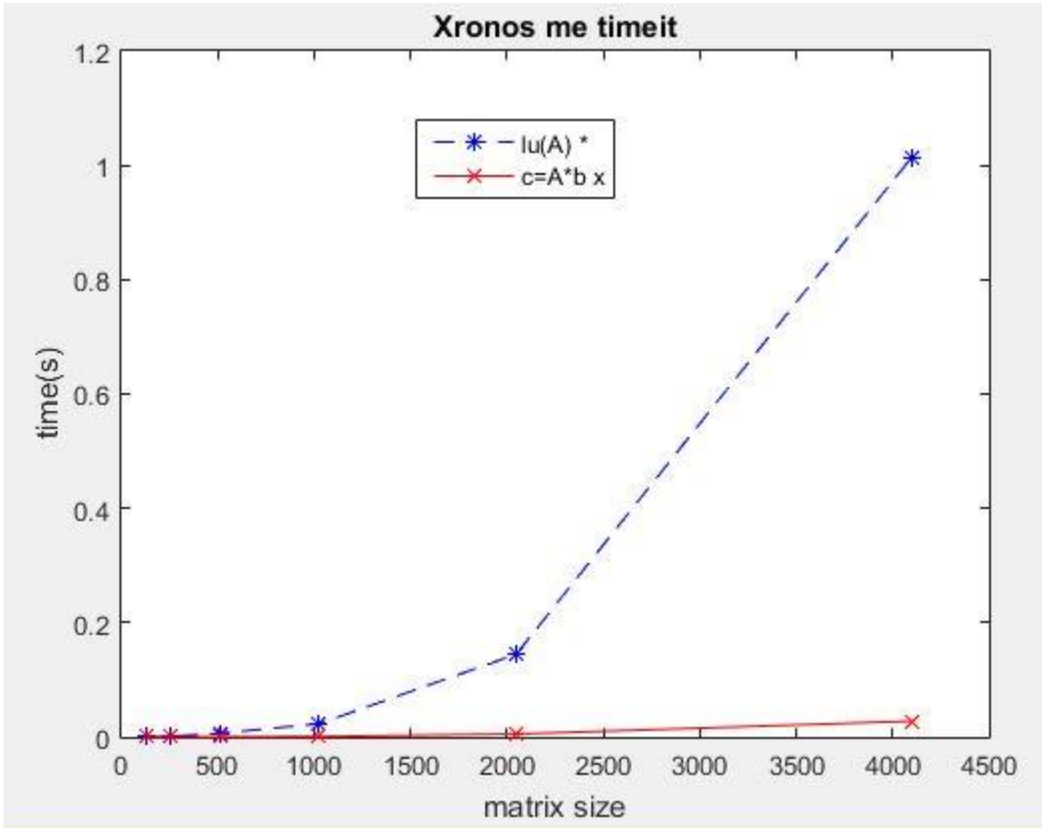
Για το (β’)


Xronos 10 ektelesewn
lu(A)
c=A*b
time(s)
matrix size
0
0.2
0.4
0.6
0.8
1
1.2
0
500
1000
1500
2000
2500
3000
3500
4000
4500

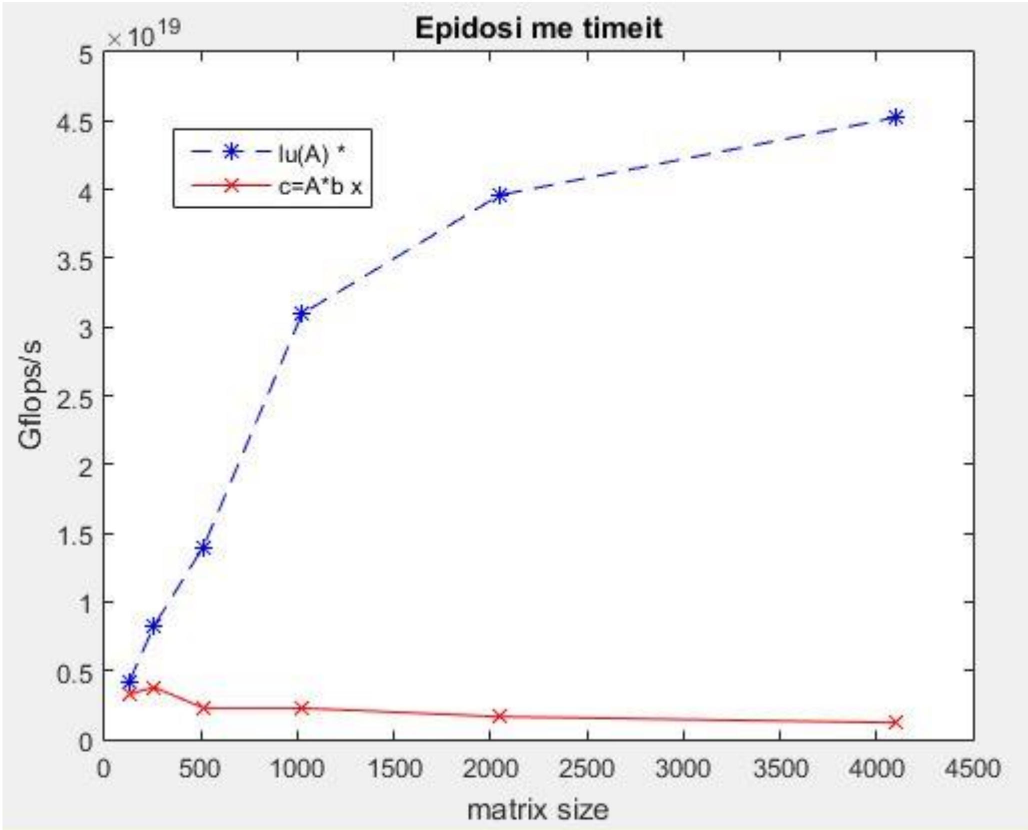


Epidosi 10 ektelesewn
×10^19
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Gflops/s
lu(A) *
c=A*b x
0
500
1000
1500
2000
2500
3000
3500
4000
4500
matrix size

Για το (γ')



Xronos me timeit
lu(A) *
c=A*b x
time(s)
matrix size
0
0.2
0.4
0.6
0.8
1
1.2
0
500
1000
1500
2000
2500
3000
3500
4000
4500




## Ερώτημα 3 – Αξιολόγηση ενδογενών συναρτήσεων και m-functions

(α) Για το κάθε ερώτημα δημιουργήθηκαν τα μητρώα με την χρήση της συνάρτησης rand. Τα αποτελέσματα αποθηκεύτηκαν στα βοηθητικά μητρώα T για τους χρόνους (με την timeit) και F για τα flops. Κάθε υπο-ερώτημα επιστρέφει τις μετρήσεις μας σε κάθε γραμμή (π.χ το πρώτο στην πρώτη γραμμή του T και F, το δεύτερο στην δεύτερη κ.ο.κ), κάθε στήλη είναι τα διαφορετικά μεγέθη μητρώων και αποθηκεύονται στο Workspace της MATLAB. Ο αριθμός των πράξεων flops είναι ίδιος για όλα τα υπο-ερωτήματα καθώς χρησιμοποιούμε την mtimes για τον πολλαπλασιασμό μητρώων.

(i) Script (E3ai)

```
for i=7:12


   n=2.^i;

   A=rand(n,n); %dhmiourgia tuxaiwn mhtrwwn

   B=rand(n,n);


   f=@()(A*B);

   T(1,i-6)=timeit(f);

   flops= 2*n^3-n^2;

   F(1,i-6)= flops/T(1,i-6)*10^9;

end
```

Με την χρήση των A=rand(n,n)  B=rand(n,n)  φτιάχνουμε τα δύο μητρώα.

Χρόνοι:

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 3.1267e-04 | 0.0012 | 0.0072 | 0.0455 | 0.3404 | 2.3044 |

Flops:

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1.3362e+19 | 2.8088e+19 | 3.7098e+19 | 4.7144e+19 | 5.0450e+19 | 5.9634e+19 |

(ii) Script(E3aii)

```
for i=7:12


   n=2.^i;

   A= full(gallery('tridiag',n,rand(1),rand(1),rand(1))); %dhmiourgia tridiagwniwn mhtrwwn

   B= full(gallery('tridiag',n,rand(1),rand(1),rand(1)));


   f=@()(A*B);

   T(2,i-6)=timeit(f);

   flops= 2*n^3-n^2;

   F(2,i-6)= flops/T(2,i-6)*10^9;

end
```

Με την χρήση των A= full(gallery('tridiag',n,rand(1),rand(1),rand(1))) και

B= full(gallery('tridiag',n,rand(1),rand(1),rand(1))) φτιάχνουμε τα δύο μητρώα. Η gallery('tridiag',n,c,d,e) φτιάχνει ένα sparse τριδιαγώνιο τετραγωνικό μητρώο μεγέθους

n, το c δίνει τιμές στην πρώτη υπο-διαγώνιο, το d στην κύρια διαγώνιο και το e στην πρώτη υπέρ-διαγώνιο. Η συνάρτηση full κάνει το sparse μητρώο full. Με την rand(1) δίνουμε τις τυχαίες μας τιμές στις 3 διαγώνιους που θέλουμε.

Χρόνοι:

| | | | | | | |
|---|---|---|---|---|---|---|
| 2 | 2.2985e-04 | 9.2826e-04 | 0.0075 | 0.0476 | 0.3291 | 2.2847 |

Flops:

| | | | | | | |
|---|---|---|---|---|---|---|
| 2 | 1.8177e+19 | 3.6077e+19 | 3.5699e+19 | 4.5134e+19 | 5.2196e+19 | 6.0149e+19 |

(iii) Script(E3aiii)

```
for i=7:12


   n=2.^i;

   A= triu(rand(n,n)); %dhmiourgia anw trigwnikwn mhtrwwn

   B= triu(rand(n,n));



   f=@()(A*B);

   T(3,i-6)= timeit(f);

   flops= 2*n^3-n^2;

   F(3,i-6)= flops/T(3,i-6)*10^9;

end
```

Με την χρήση των A= triu(rand(n,n)) B= triu(rand(n,n)) φτιάχνουμε τα δύο άνω τριγωνικά μητρώα. Η συνάρτηση triu κάνει τα δύο τυχαία μητρώα άνω τριγωνικά.

Χρόνοι:

| | | | | | | |
|---|---|---|---|---|---|---|
| 3 | 2.1074e-04 | 9.3005e-04 | 0.0072 | 0.0431 | 0.3350 | 2.3075 |

Flops:

| | | | | | | |
|---|---|---|---|---|---|---|
| 3 | 1.9825e+19 | 3.6008e+19 | 3.7490e+19 | 4.9777e+19 | 5.1270e+19 | 5.9555e+19 |

(iv) Script(E3aiv)

```
for i=7:12


   n=2.^i;

   A= hess(rand(n,n)); %dhmiourgia mhtrwwn anw Hessenberg

   B= hess(rand(n,n));



   f=@()(A*B);

   T(4,i-6)= timeit(f);

   flops= 2*n^3-n^2;

   F(4,i-6)= flops/T(4,i-6)*10^9;

end
```

Με την χρήση των A= hess(rand(n,n)) B= hess(rand(n,n)) φτιάχνουμε τα δύο μητρώα. Η συνάρτηση hess κάνει τα τυχαία μητρώα άνω Hessenberg.

Χρόνοι:

| | | | | | | |
|---|---|---|---|---|---|---|
| 4 | 1.9484e-04 | 8.4702e-04 | 0.0069 | 0.0433 | 0.3465 | 2.3279 |

Flops:

| | | | | | | |
|---|---|---|---|---|---|---|
| 4 | 2.1442e+19 | 3.9537e+19 | 3.9026e+19 | 4.9573e+19 | 4.9565e+19 | 5.9032e+19 |

(β) Ο κώδικας της myTridMult είναι:

```
function [G]= myTridMult(matrix1,matrix2)


  n= length(matrix1);

  A= matrix1;

  B= matrix2;

  G= zeros(n,n); %neo mhtrwo me to ginomeno twn A,B


  G(1,1)=A(1,1)*B(1,1)+A(1,2)*B(2,1); % arxiko kai teliko stoixeio ths kurias diagwniou

  G(n,n)=A(n,n-1)*B(n-1,n)+A(n,n)*B(n,n);


  for i=1:n-1


    G(i,i+1)=A(i,i)*B(i,i+1)+A(i,i+1)*B(i+1,i+1); %stoixeia stin uper-diagwnio

    G(i+1,i)=A(i+1,i)*B(i,i)+A(i+1,i+1)*B(i+1,i); %stoixeia stin upo-diagwnio


    if i <= n-2 % stoixeia ektos 3 diagwniwn

    G(i,i+2)=A(i,i+1)*B(i+1,i+2);

    G(i+2,i)=A(i+2,i+1)*B(i+1,i);

    end


    if i>1 %stoixeia kurias diagwniou ektws (1,1) kai (n,n)
```

```
        G(i,i)=A(i,i-1)*B(i-1,i)+A(i,i)*B(i,i)+A(i,i+1)*B(i+1,i);
        end
    end
end
```

Δημιουργήθηκε η συνάρτηση myTridMult, η οποία επιστρέφει το γινόμενο δύο τριδιαγωνικών μητρώων. Χρησιμοποιήθηκε το Mathematica ώστε να βρούμε με κάποια μικρά παραδείγματα τα συνήθη στοιχεία του παραγώμενου πίνακα που παίρνουν τιμές διάφορες του μηδενός. Η συνάρτηση δέχεται σαν ορίσματα δύο τριδιαγώνιους πίνακες και επιστρέφει ένα νέο μητρώο με το γινόμενό τους. Η διαφορά της σε σχέση με την mtimes(A,B) είναι οτι η δικιά μας συνάρτηση αποφεύγει τις πράξεις με τα μηδενικά, συγκεντρώνεται μόνο σε αυτές που καταχωρούν τιμή στο νέο μητρώο διαφορετική του μηδενός. Η συγκεκριμένη συνάρτηση θα επιστρέψει σωστό μητρώο αν και μόνο αν δωθούν ως ορίσματα δύο τριδιαγωνικά μητρώα. Στο βοηθητικό μητρώο C, αποθηκεύουμε τους χρόνους των συναρτήσεων για διαφορετικά μεγέθη μητρώων. Όπως φαίνεται παρακάτω η myTridMult “κερδίζει” αρκετό χρόνο σε μητρώα μεγάλου μεγέθους καθώς αποφεύγει πάρα πολλές πράξεις. Μετρήσαμε τον αριθμό των πράξεων που κάνει η myTridMult μέσα στον βρόγχο και εκτός αυτού.

Χρόνοι ( $1^η$ γραμμή mtimes, $2^η$ myTridMult):

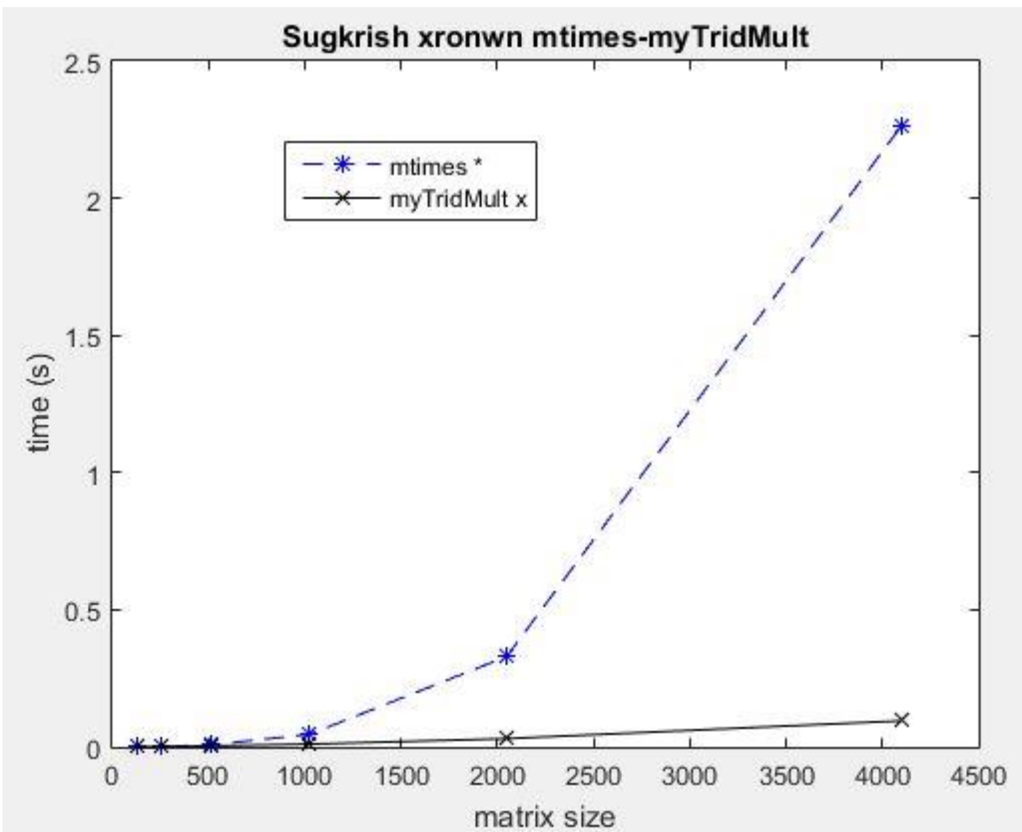
| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 2.1768e-04 | 0.0011 | 0.0076 | 0.0458 | 0.3291 | 2.2598 |
| 2 | 8.4205e-04 | 0.0016 | 0.0040 | 0.0103 | 0.0304 | 0.0951 |

(Script E3b) Για την σύγκριση χρόνου/επίδοσης πολλαπλασιασμού τριδιαγώνιων μητρώων:

```
        for i=7:12

  n=2.^[i];

  A= full(gallery('tridiag',n,rand(1),rand(1),rand(1))); %dimiourgia tridiagwniwn mhtrwwn
  B= full(gallery('tridiag',n,rand(1),rand(1),rand(1)));

  f=@()(A*B);
  C(1,i-6)= timeit(f);

  f=@()myTridMult(A,B);
  C(2,i-6)= timeit(f);

end

n=2.^[7:12];

figure %xronoi
plot(n,C(1,:),'b*--')
hold on
plot(n,C(2,:),'kx-')
```

```
hold off

legend('mtimes *','myTridMult x')

title('Sugkrish xronwn mtimes-myTridMult')

xlabel('matrix size')

ylabel('time (s)')
```

Συγκριτική γραφική παράσταση:



(γ) Για να παρουσιάσουμε μια κοινή γραφική παράσταση για τους χρόνους και μια άλλη για τα Gflops/s δημιουργήθηκε ένα καινούριο script (E3Diagrams).

```
        for i=[7:12]

  n=2.^[i];

  A= full(gallery('tridiag',n,rand(1),rand(1),rand(1))); %dhmiourgia tridiagwniwn mhtrwwn
  B= full(gallery('tridiag',n,rand(1),rand(1),rand(1)));

  f=@()myTridMult(A,B);
  T(5,i-6)= timeit(f);
  flops= 13*n-14;
  F(5,i-6)= flops/T(5,i-6)*10^9;

end

n=2.^[7:12];

figure %Xronoi
plot(n,T(1,:),'g*--')
hold on
plot(n,T(2,:),'rx-')
```

```
hold on

plot(n,T(3,:),'bd:')

hold on

plot(n,T(4,:),'ys-.')

hold on

plot(n,T(5,:),'kh--')

hold off

legend('rand *','tridiag x','triau d','hess square','tridmult hexagramm')

title('Xronoi mult A,B me mtimes')

xlabel('matrix size')

ylabel('time (s)')


figure %Gflops

plot(n,F(1,:),'g*--')

hold on

plot(n,F(2,:),'rx-')

hold on

plot(n,F(3,:),'bd:')

hold on

plot(n,F(4,:),'ys-.')

hold on

plot(n,F(5,:),'kh--')
```

hold off

legend('rand *','tridiag x','triau d','hess square','tridmult hexagramm')

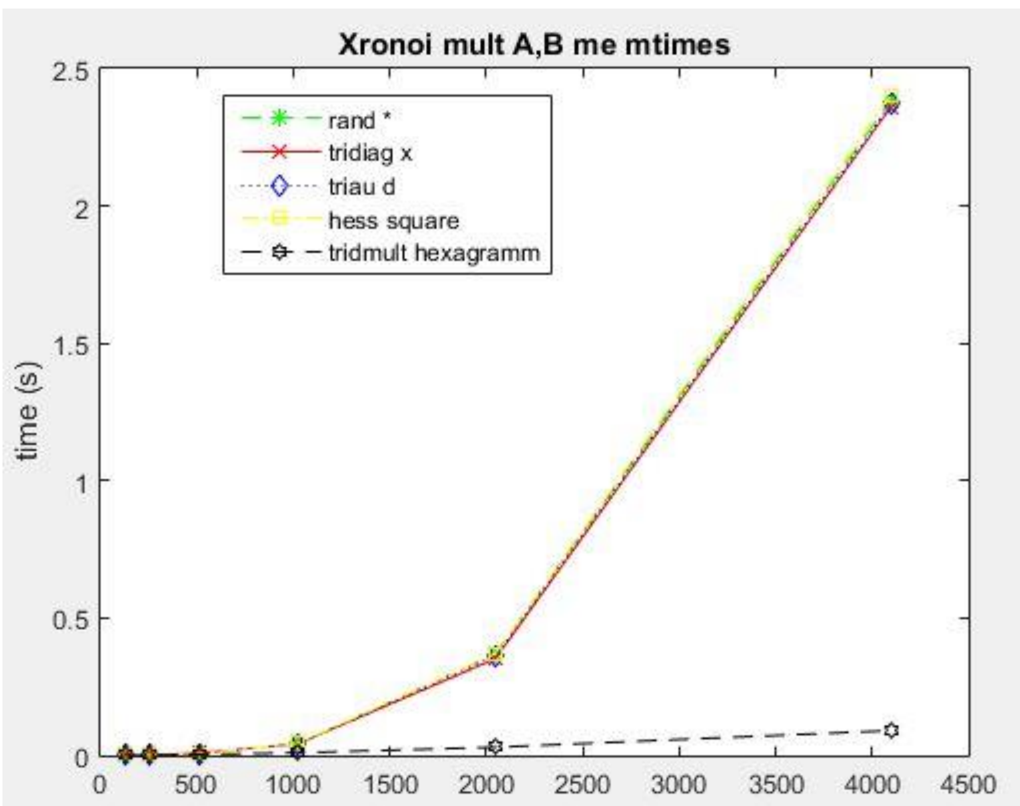
title('Epidosi mult A,B me mtimes')

xlabel('matrix size')

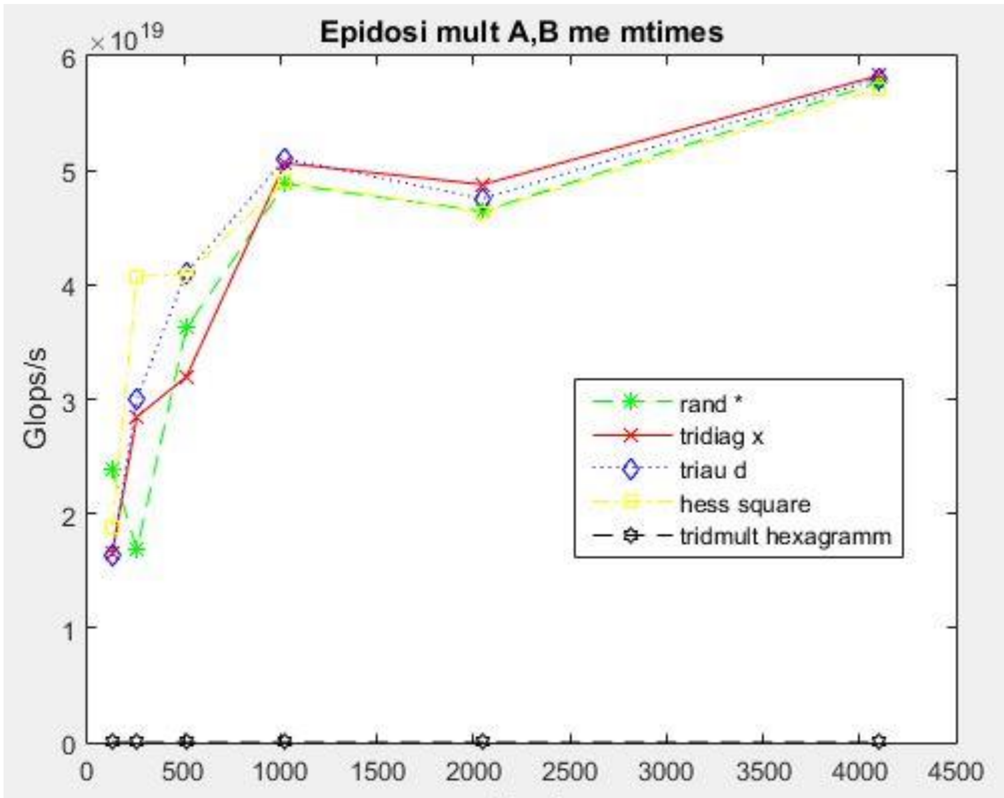
ylabel('Glops/s')

Τα συγκριτικά διαγράμματα είναι τα εξής:

Χρόνοι:



Flops:




(δ) Είναι λογικό να υπάρχουν κάποιες μικροδιαφορές στις επιδόσεις. Αυτό συμβαίνει επειδή εκτός από την myTridMult , ολά τα άλλα έχουν κοινό αριθμητή αλλά διαφορετικό παρονομαστή, ανάλογα με την καθυστέρηση του κάθε μητρώου. Όσον αφορά την myTridMult , έχει πολλά λίγα Glops/s καθώς έχεις σημαντικά λιγότερο αριθμό πράξεων σε σχέση με την mtimes.

## Ερώτημα 4 – Σύγκριση Υλοποιήσεων

((u*u.’ + v*v.’)^p)*b

(1) Υπολογίστηκε το απαιτούμενο πλήθος πράξεων συναρτήσει των n και p. Για να γίνει αυτό, μετρήσαμε τον αριθμό των πράξεων που γίνονται:

μέσα στην παρένθεση: 3*n^2

του μητρώου στην δύναμη p: (p-1)(2*n^3-n^2)

του μητρώου επί διάνυσμα στήλη: 2*n^2-n

και στην συνέχεια τους προσθέσαμε και καταλήξαμε στο: 2*(p-1)*n^3+(6-p)*n^2-n.

(2) (function rank2_power)

```
function [x] = rank2_power(v1,v2,v3)


u=v1;

v=v2;

b=v3;


if (~iscolumn(u) || ~iscolumn(v) || ~iscolumn(b)) %elegxos orismatwn

   error('Inputs must be vectors')

end


   L= length(u); %kataxwroume sto L tin timh n

   L1= length(v);

   L2= length(b);
```

```
    if L==L1 && L1==L2 %elegxos isothtas megethous orismatwn

        x= (u*u.' + v*v.')^10 *b; %prakseis parastashs me proteraiothta

    end
end
```

Για p=10 δημιουργήθηκε η συνάρτηση rank2_power η οποία δέχεται σαν ορίσματα τρια διανύσματα στήλες, υπολογίζει την παράσταση του ερωτήματς και επιστρέφει ένα νέο διάνυσμα με το αποτέλεσμα της παράστασης τηρώντας την προτεραιότητα τον πράξεων από αριστερά προς τα δεξιά.

(3) (function my_rank2_power)

```
function [x] = my_rank2_power(v1,v2,v3)


u=v1;
v=v2;
b=v3;


if (~iscolumn(u) || ~iscolumn(v) || ~iscolumn(b)) %elegxos orismatwn
    error('Inputs must be vectors')
end


    L= length(u); %kataxwroume sto L tin timh n
```

```
    L1= length(v);

    L2= length(b);


    if L==L1 && L1==L2 %elegxos isothtas megethous orismatwn


        A= (u*u.' + v*v.'); %apothikeusi ths parenthesis mesa se ena mhtrwo A gia voitheia

        x= A*(A*(A*(A*(A*(A*(A*(A*(A*(A*b))))))))); %prakseis parastashs meta thn metatroph


    end

end
```

Για p=10 δημιουργήθηκε η συνάρτηση my_rank2_power η οποία δέχεται σαν ορίσματα τρια διανύσματα στήλες βγάζει το ίδιο αποτέλεσμα με την rank2_power αλλά με σημαντικά μικρότερο Ω. Για να επιτευχθεί αυτό, μετατρέξαμε την παράσταση με τρόπο τέτοιο ώστε να έχουν άλλη σειρά προτεραιότητας οι πράξεις. Για ευκολία ονομάσαμε την παρένθεση, έστω A. Εφόσον έγινε ο πολλαπλασιασμός g= A*b, στην συνέχεια πολλαπλασιάσαμε άλλες εννιά φορές το g= A*g όπου g είναι κάθε φορά το γινόμενο μητρώο (στην περίπτωσή μας διάνυσμα) του προηγούμενου πολλαπλασιασμού. Ο λόγος που μειώθηκε σημαντικά το Ω, είναι επειδή κάθε φορά γίνεται ο πολλαπλασιασμός μητρώο επί διάνυσμα που μας παράγει διάνυσμα, σε αντίθεση με την rank2_power που κάνει τον πολλαπλασιασμό μητρώο επί μητρώο που παράγει μητρώο.

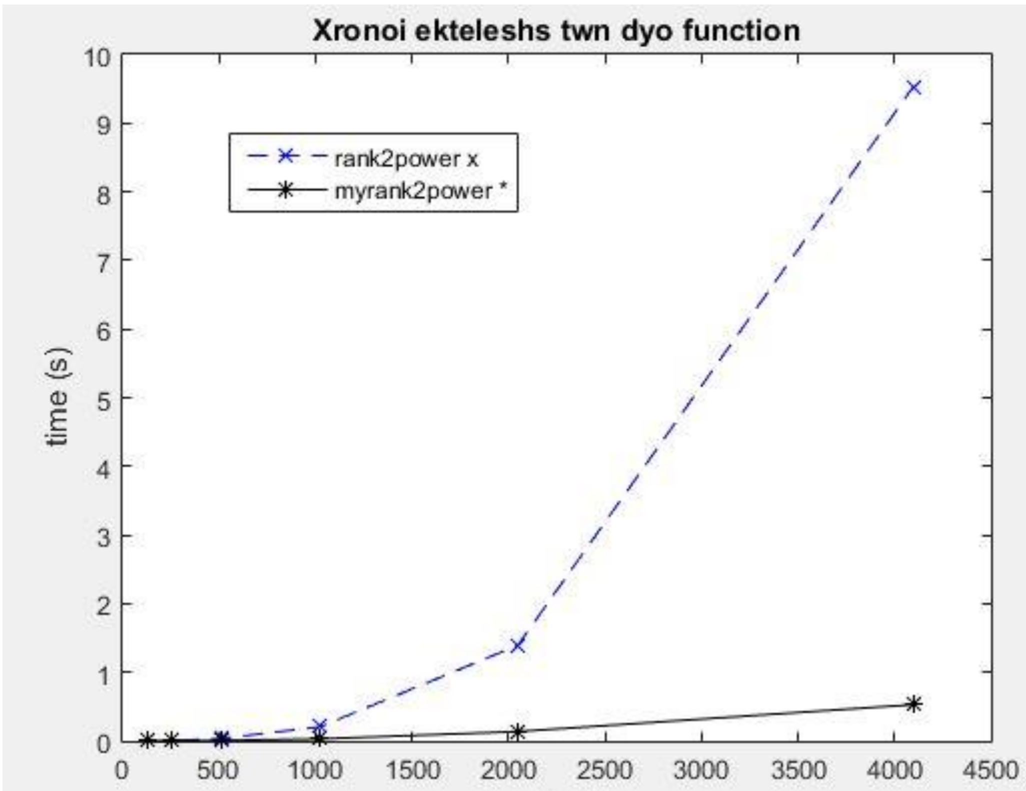
(4) (Script E44Diagrams)

```
for i=[7:12]


n= 2.^[i];


   u= rand(n,1);

   v= rand(n,1);

   b= rand(n,1);


   f=@() rank2_power(u,v,b);

   T(1,i-6)= timeit(f);

   flops= 18*n^3 -4*n^2-n;

   F(1,i-6)= flops/T(1,i-6)*10^9;


   f=@() my_rank2_power(u,v,b);

   T(2,i-6)= timeit(f);

   flops= 23*n^2 -10*n;

   F(2,i-6)= flops/T(2,i-6)*10^9;


end
```

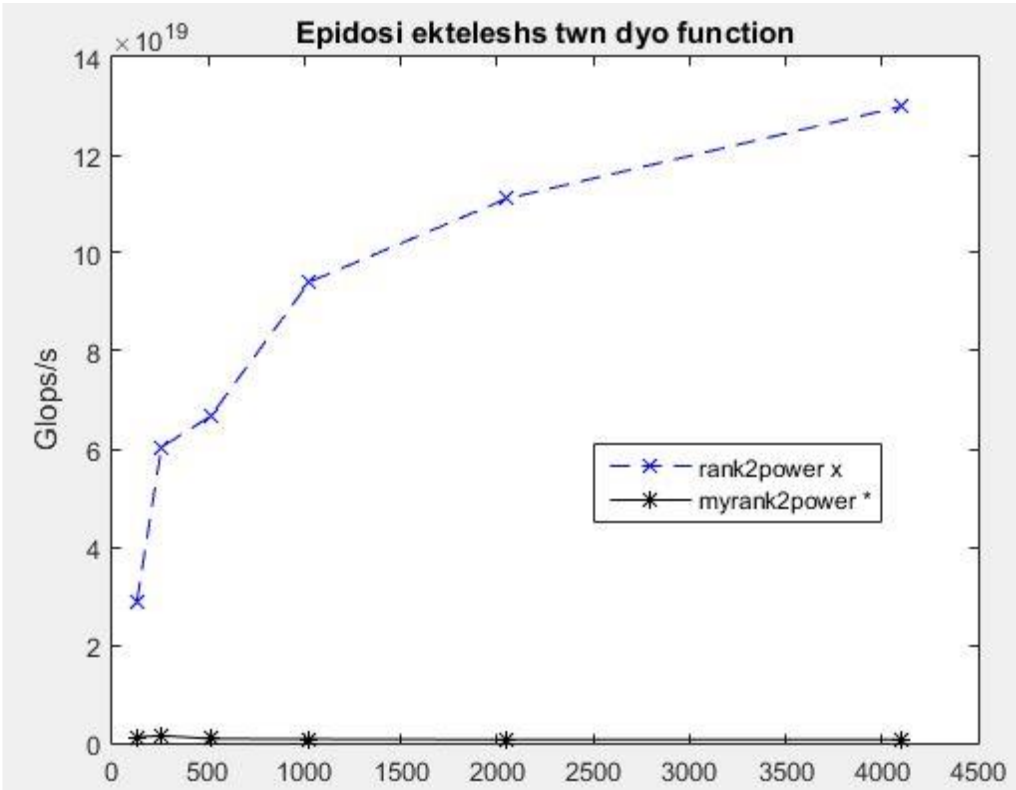
```
n=2.^[7:12];

figure %Xronoi
plot(n,T(1,:),'bx--')
hold on
plot(n,T(2,:),'k*-')
hold off
legend('rank2power x', 'myrank2power *')
title('Xronoi ekteleshs twn dyo function')
xlabel('matrix size')
ylabel('time (s)')

figure %Epidosi
plot(n,F(1,:),'bx--')
hold on
plot(n,F(2,:),'k*-')
hold off
legend('rank2power x','myrank2power *')
title('Epidosi ekteleshs twn dyo function')
xlabel('matrix size')
ylabel('Glops/s')
```

Για n=2.^[7:12] μετρήθηκε η επίδοση των δύο συναρτήσεων. Δημιουργήσαμε ένα νέο script το οποίο κάνει την σύγκριση αυτή:

(α)



(β)




Είναι σημαντικό να σημειωθεί ότι ο αριθμός των πράξεων Ω της rank2_power για την συγκεκριμένη περίπτωση είναι :

| | 1 |
|---|---|
| 1 | 1.2369e+12 |

ενώ της my_rank2_power είναι:

| | 1 |
|---|---|
| 1 | 385835008 |