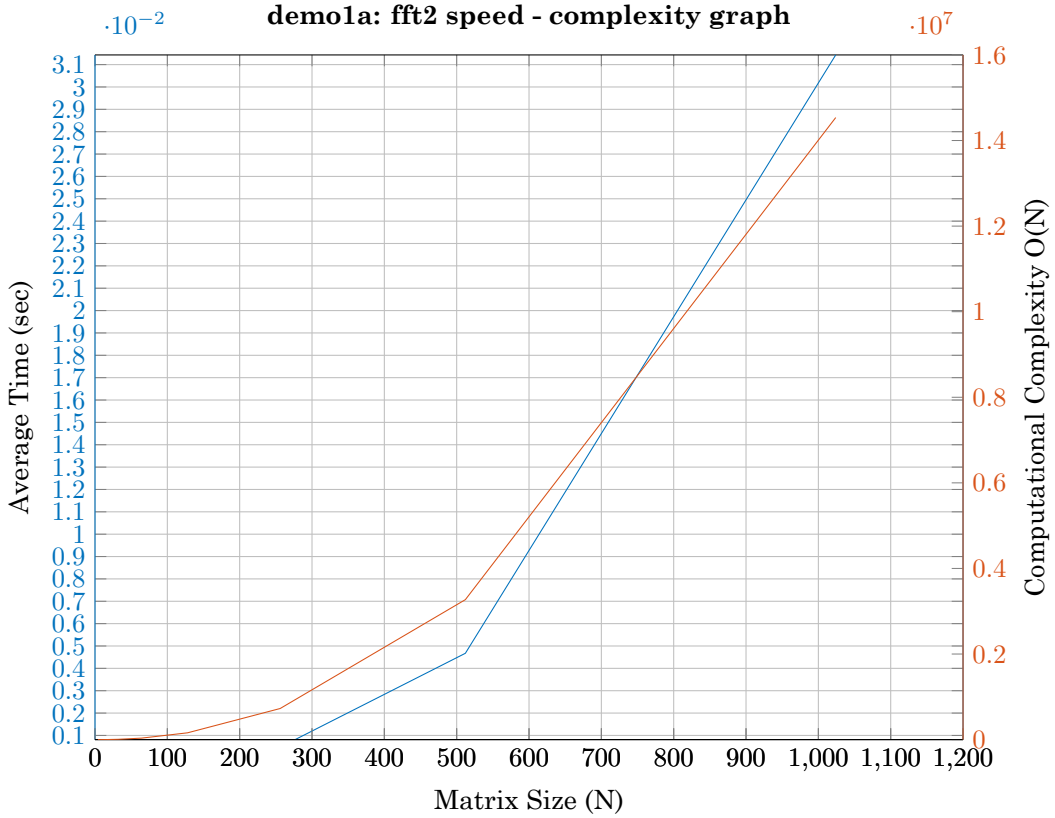
ΨΕΕ: Εργασία #1

Δημανίδης Τάσος - 7422

Περιεχόμενα

Ζητούμενο 1

Με την εκτέλεση του demo1a.m εμφανίζεται το παρακάτω γράφημα:



Η μπλε γραμμή αντιστοιχεί στην ταχύτητα εκτέλεσης του fft2 και η πορτοκαλί στην αντίστοιχη θεωρητική πολυπλοκότητα. Κατ' αρχήν παρατηρούμε ότι οι δύο καμπύλες είναι όμοιες. Η αύξηση της πολυπλοκότητας έχει ως αποτέλεσμα ανάλογη αύξηση στην ταχύτητα εκτέλεσης. Επίσης παρατηρούμε ότι όσο μεγαλώνει το μέγεθος του πίνακα, τόσο ο χρόνος εκτέλεσης όσο και η πολυπλοκότητα αυξάνουν εκθετικά. Αυτό είναι λογικό δεδομένου ότι το μέγεθος του πίνακα είναι N^2

Ζητούμενο 2

Ζητούμενο 2.1

Αρχικά συγγράφηκε ο παρακάτω κώδικας

```
function z = myconv2(h,x)

[MX NX] = size(x);
[MH NH] = size(h);
MZ = MX + MH - 1;
NZ = NX + NH - 1;

x = padarray(x,[MH-1 NH-1]);
h = flip(flip(h,2),1);
h = h(:)';

z = zeros(MZ,NZ);

for K1 = 1:MZ
        h_range = K1:((K1-1)+MH);
        for K2 = 1:NZ
                x_range = K2:((K2-1)+NH);
                xs = x(h_range,x_range);
                z(K1,K2) = h*xs(:);
        end
end
```

Εδώ αρχικά συμπληρώνεται το σήμα x με 0 και η μάσκα υπόκειται flip. Η ανάθεση των επιμέρους αθροισμάτων, στον πίνακα συνέλιξης z, γίνεται με διπλό for. Κατά την εκτέλεση του διπλού for πολλαπλασιάζουμε την μάσκα με έναν κινούμενο υποπίνακα του σήματος x. Ο υποπίνακας αυτός έχει τις διαστάσεις της μάσκας και κινείται από αριστερά προς τα δεξιά και από επάνω προς τα κάτω.
Η εκτέλεση αυτής της συνάρτησης με τα σήματα του data.mat δίνει έξοδο:

```
>> myconv2_old(y,x)

ans =

    3.4655   16.4916   30.9824   27.8706    9.3879
    5.0339   26.1971   49.2425   47.6631   19.5990
   22.4419   29.3474   31.7719   27.9956   10.9564
   10.6887   22.5403    7.7466    9.2875    1.7925
```

Το Ζητούμενο 2 συνεχίζεται στην επόμενη σελίδα…

Ο κώδικας αυτός βελτιώθηκε και έγινε συνολικό vectorization της υλοποίησης. Το αποτέλεσμα βρίσκεται στο αρχείο myconv2.m της εργασίας:

```
function z = myconv2(h,x)

[MX NX] = size(x);
[MH NH] = size(h);
MZ = MX + MH - 1;
NZ = NX + NH - 1;

x = padarray(x,[MH-1 NH-1]);
[MXP NXP] = size(x);
h = h';
h = fliplr(h(:)');

% generate xp index base vectors
xpC = 1:MXP:((NH-1)*MXP+1);
xpCE = bsxfun(@plus,xpC(:),0:(MH-1)); % periodically extended column
xpR = 0:MXP:((NZ-1)*MXP);
xpRE = bsxfun(@plus,xpR(:),0:(MZ-1)); % periodcially extended row

% generate xp index grid from the base vectors
xpIDX = bsxfun(@plus,xpCE(:),xpRE(:)');

z = vec2mat(h*x(xpIDX),NZ);
```

Η ιδέα εδώ είναι να πολλαπλασιάσουμε 1 πίνακα γραμμή (την μάσκα) με 1 πίνακα με N στήλες. Με θα αναφερόμαστε στο γινόμενο των διαστάσεων του πίνακα συνέλιξης. Κάθε μία από αυτές τις στήλες θα αντιστοιχεί σε 1 υποπίνακα, ο οποίος θα είναι όπως περιγράψαμε προηγουμένως. Το αποτέλεσμα θα είναι πολλαπλασιασμοί πινάκων. Στο τέλος ο πίνακας $1xN$ θα αναδιοργανώνεται ώστε να πάρει τις διαστάσεις του σήματος συνέλιξης.
Προκειμένου να το πετύχουμε αυτό, δημιουργούμε πρώτα έναν πίνακα που θα περιέχει τα index των υποπινάκων που αντιστοιχούν στο σήμα x. Ο πίνακας αυτός, $xpIDX$, είναι προφανώς και αυτός μεγέθους $1xN$. Κάθε στήλη του περιέχει τα index του i-οστού υποπίνακα. Οι στήλες αυτές είναι εύκολο να δημιουργηθούν όταν διαπιστώσουμε τα εξής:

1. Η Matlab μπορεί να προσπελάσει δισδιάστατο πίνακα με μονό index. Το index αυτό αυξάνει πρώτα κατά γραμμή και μετά κατά στήλη.
2. Τα στοιχεία του υποπίνακα συνέλιξης πολλαπλασιάζονται από αριστερά προς δεξιά και από πάνω προς τα κάτω με την flipped μάσκα. Άρα έχουν περιοδικότητα που σχετίζεται με τις διαστάσεις της μάσκας. Μάλιστα για να πάμε από την πάνω γραμμή στην κάτω, το μόνο που κάνουμε είναι να αυξήσουμε τα index της πάνω κατά 1 (πάντα με βάση το indexing της Matlab όπως περιγράφηκε στο 1.). Τα στοιχεία αυτά δεν είναι άλλα από τις γραμμές κάθε στήλης του πίνακα $xpIDX$
3. Ο δεξιότερος ενός υποπίνακα συνέλιξης έχει τα ίδια index με τον αριστερό (δηλαδή τον προηγούμενο) +1. Επίσης μόλις φτάσουμε στον πιο δεξιό υποπίνακα συνέλιξης, καταβαίνουμε μία γραμμή κάτω και ξεκινάμε από την πιο αριστερή θέση. Στην νέα αυτή θέση, ο νέος υποπίνακας προφανώς δεν είναι τα index του προηγούμενου +1. Είναι όμως τα index του προηγούμενου πιο αριστερού υποπίνακα +1. Άρα και εδώ υπάρχει περιοδικότητα, η οποία σχετίζεται με το πόσες φορές κινείται ο υποπίνακας προς τα δεξιά. Δηλαδή με το NZ, τις στήλες του σήματος συνέλιξης.

Το Ζητούμενο 2 συνεχίζεται στην επόμενη σελίδα…

Με βάση αυτές τις παρατηρήσεις δημιουργούμε πρώτα ένα base vector στήλη, $xpCE$, το οποίο υλοποιεί το 2. με βάση το 1.. Στην συνέχεια δημιουργούμε έναν base vector γραμμή, $xpRE$, που αποτυπώνει την λογική του 3. πάλι με βάση το 1.. Τέλος παίρνουμε αυτά τα δύο base vectors και "εφαρμόζουμε την γραμμή στην στήλη". Δηλαδή την λογική εξέλιξης της γραμμής την αξιοποιούμε για να δημιουργήσουμε τις στήλες. Το αποτέλεσμα είναι ο πίνακας των index, $xpIDX$.

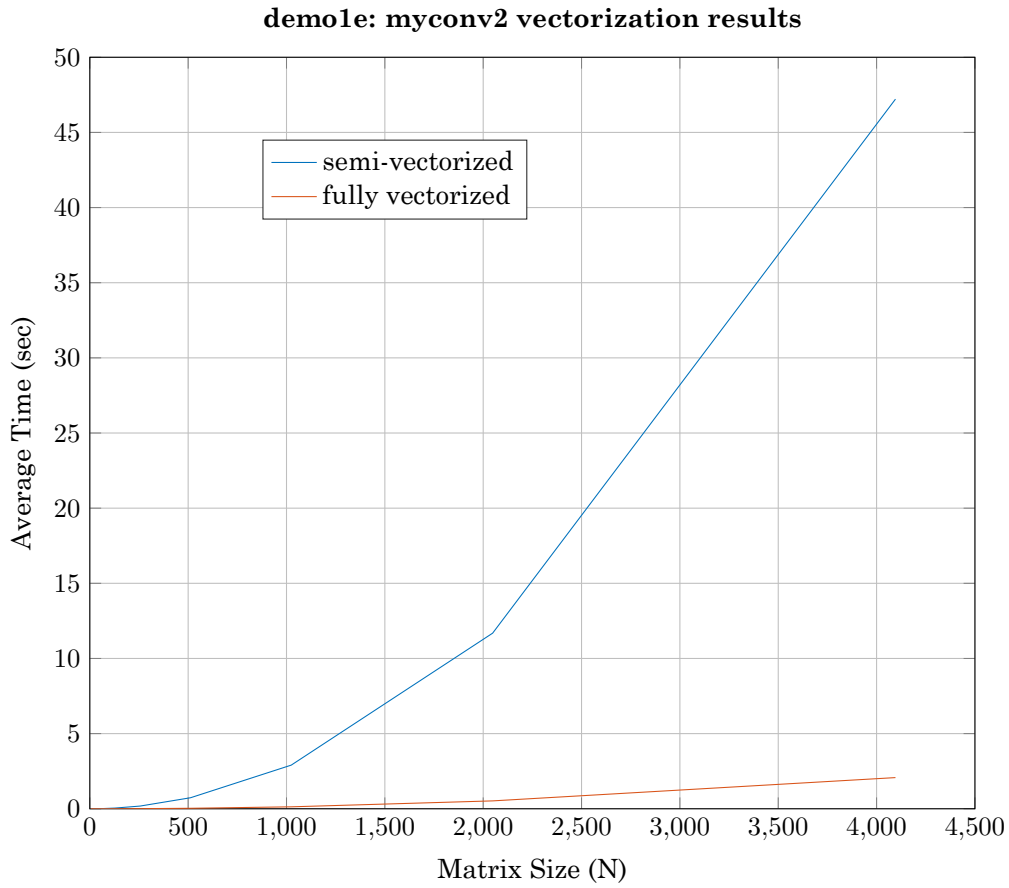
Το μόνο που μένει είναι να πολλαπλασιάσουμε την μάσκα με αυτόν τον πίνακα. Το αποτέλεσμα αυτού του πολλαπλασιασμού πινάκων, είναι όπως είπαμε ένας πίνακας $1xN$ τον οποίο αναδιοργανώνουμε στις διαστάσεις του πίνακα συνέλιξης με το vec2mat.
Η εκτέλεση αυτής της συνάρτησης με τα σήματα του data.mat δίνει έξοδο:

```
>> myconv2(y,x)

ans =

    3.4655   16.4916   30.9824   27.8706    9.3879
    5.0339   26.1971   49.2425   47.6631   19.5990
   22.4419   29.3474   31.7719   27.9956   10.9564
   10.6887   22.5403    7.7466    9.2875    1.7925
```

Προκειμένου να διαπιστωθεί η διαφορά στην απόδοση, συγγράφηκε το script demo1e.m. Το script αυτό εκτελεί το πείραμα με τις 100 επαναλήψεις, για k έως και 12 αντί 10, για τις δύο συναρτήσεις. Η εκτέλεση του script είχε το παρακάτω αποτέλεσμα:



Φαίνεται ξεκάθαρα ότι το ολικό vectorization βελτίωσε δραματικά την επίδοση του κώδικα.

Το Ζητούμενο 2 συνεχίζεται στην επόμενη σελίδα…

Ζητούμενο 2.2

Το αρχείο myconv2freq.m έχει ως εξής

```
function z = myconv2freq(h,x)

[MX NX] = size(x);
[MH NH] = size(h);

xp = padarray(x,[MH NH], 'post');
hp = padarray(h,[MX NX], 'post');

M = MX + MH;
N = NX + NH;

u1n1 = (0:(M-1))'*(0:(M-1));
u2n2 = (0:(N-1))'*(0:(N-1));
WN1 = exp(i*(2*pi*u1n1)/M);
WN2 = exp(i*(2*pi*u2n2)/N);
X = (WN1)'*(xp)*conj(WN2);
H = (WN1)'*(hp)*conj(WN2);

Z = X.*H;
z = (1/(M*N))*WN1*(Z)*((WN2).');
```

Η υλοποίηση του μετασχηματισμού fourier βασίζεται στις σελίδες 8,9,10 της διαφάνειας 6. Δηλαδή στην δημιουργία ενός πίνακα W_N ώστε με την αξιοποίηση των τύπων να προκύψει ο FT. Αρχικά κάνουμε pad τα σήματα x,h και στην συνέχεια δημιουργούμε τους πίνακες W_{N1}, W_{N2} (οι οποίοι είναι ίδιοι αν τα σήματα είναι NxN). Στην συνέχεια με βάση τους τύπους, υπολογίζουμε τους FT της μάσκας και του σήματος. Έπειτα πολλαπλασιάζουμε ανά στοιχείο (όχι πίνακα με πίνακα) τους δύο αυτούς FT και παίρνουμε την συνέλιξη στο πεδίο της συχνότητας. Τέλος, προκειμένου να επιστρέψουμε στο πεδίο του χρόνου, εφαρμόζουμε τον IFT πάλι με βάση τον τύπο με τα W_N και κανονικοποιούμε το αποτέλεσμα.
Η εκτέλεση αυτής της συνάρτησης με τα σήματα του data.mat δίνει έξοδο:

```
ans =

  Columns 1 through 5

   3.4655 - 0.0000i  16.4916 - 0.0000i  30.9824 - 0.0000i  27.8706 - 0.0000i   9.3879 + 0.0000i
   5.0339 - 0.0000i  26.1971 - 0.0000i  49.2425 - 0.0000i  47.6631 - 0.0000i  19.5990 + 0.0000i
  22.4419 - 0.0000i  29.3474 - 0.0000i  31.7719 - 0.0000i  27.9956 - 0.0000i  10.9564 + 0.0000i
  10.6887 + 0.0000i  22.5403 + 0.0000i   7.7466 + 0.0000i   9.2875 + 0.0000i   1.7925 - 0.0000i
  -0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i

  Column 6

   0.0000 - 0.0000i
   0.0000 - 0.0000i
   0.0000 - 0.0000i
  -0.0000 - 0.0000i
  -0.0000 - 0.0000i
```

Ζητούμενο 3

Ζητούμενο 3.1

Θεωρούμε την πολυπλοκότητα των pad,flip αμελητέα.

1. Η παραγωγή του $xpCE$ απαιτεί $MH * NH$ πράξεις, άρα 4 για σταθερό h
2. Η παραγωγή του $xpRE$ απαιτεί $MZ * NZ$ πράξεις
3. Η παραγωγή του $xpIDX$ απαιτεί $4 * MZ * NZ$ πράξεις
4. Ο πολλαπλασιασμός πινάκων είναι συνολικά $4 * 4 * MZ * NZ$ πράξεις
4. Η αναδιαμόρφωση του πίνακα απαιτεί NZ πράξεις

άρα η συνολική θεωρητική πολυπλοκότητα της myconv2:

$O = MZ * NZ + 4 * MZ * NZ + NZ + 4 * 4 * MZ * NZ = O(N^2)$

Ζητούμενο 3.2

Και εδώ θεωρούμε την πολυπλοκότητα του pad αμελητέα.

1. Η παραγωγή των διανυσμάτων $u1n1$, $u2n2$ απαιτεί M^2 και N^2 πράξεις αντίστοιχα
2. Η παραγωγή των W_{N1}, W_{N2} απαιτεί M και N πράξεις αντίστοιχα
3. Οι FT απαιτούν $M * N * N$ πράξεις
4. Η συνέλιξη στο πεδίο της συχνότητας $*$ πράξεις
5. Η επιστροφή στο πεδίο του χρόνου απαιτεί πάλι $M * N * N$ πράξεις

άρα η συνολική θεωρητική πολυπλοκότητα της myconv2freq:

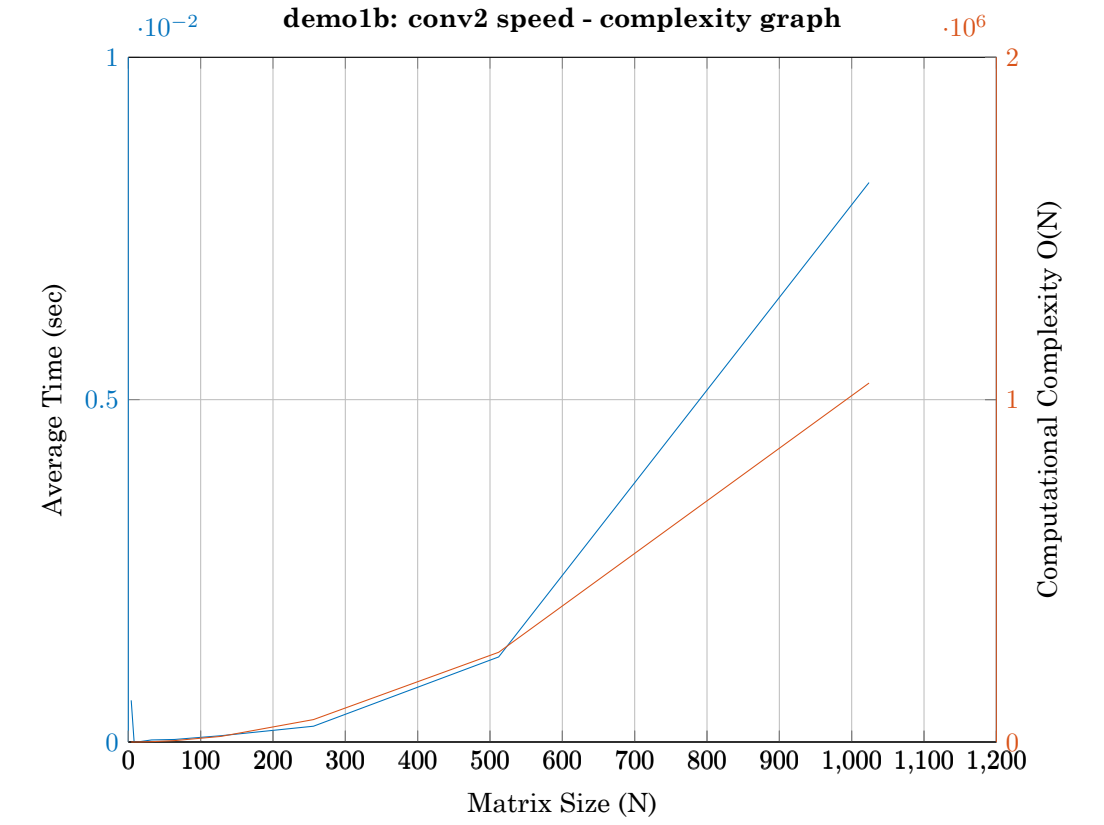
$O = M^2 + N^2 + M + N + M * N * N + M * N + M * N * N = O(N^3)$

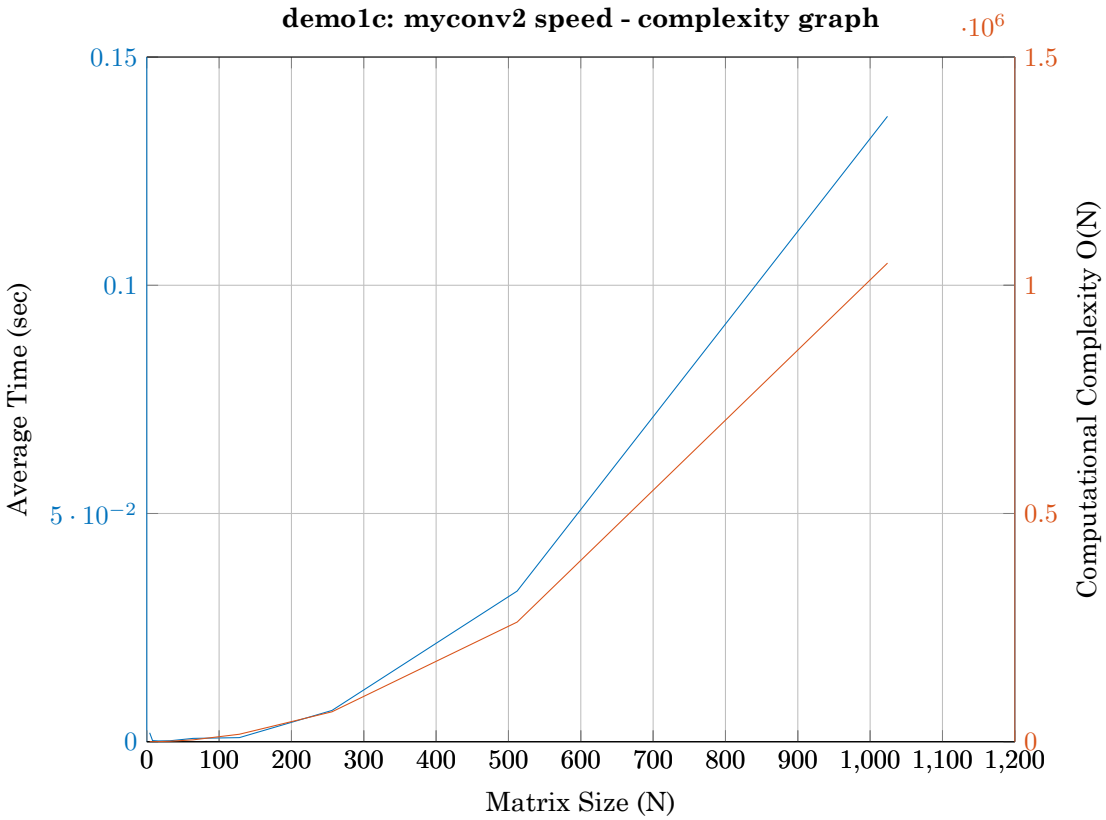
Με βάση την θεωρία η πολυπλοκότητα του FFT είναι $O(N^2 * log(N))$ για $M = N$

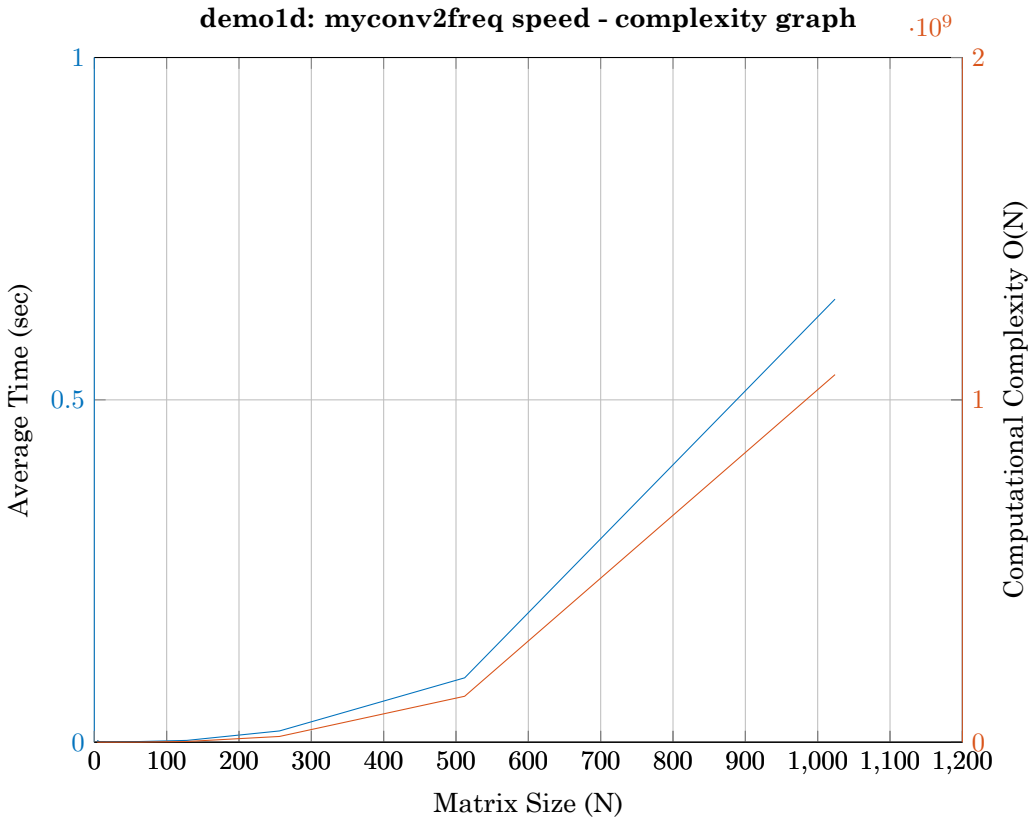
Ζητούμενο 3.3

Παρακάτω φαίνονται τα γραφήματα που προέκυψαν από τις εκτελέσεις των script demo1b,demo1c,demo1d

Το Ζητούμενο 3 συνεχίζεται στην επόμενη σελίδα…


demo1b: conv2 speed - complexity graph
$\cdot 10^{-2}$
$\cdot 10^{6}$
1
0.5
0
2
1
0
Average Time (sec)
Computational Complexity O(N)
0 100 200 300 400 500 600 700 800 900 1,000 1,100 1,200
Matrix Size (N)


demo1c: myconv2 speed - complexity graph
$\cdot 10^{6}$
0.15
0.1
$5 \cdot 10^{-2}$
0
1.5
1
0.5
0
Average Time (sec)
Computational Complexity O(N)
0 100 200 300 400 500 600 700 800 900 1,000 1,100 1,200
Matrix Size (N)



Παρατηρούμε ότι η συνάρτηση της Matlab είναι η γρηγορότερη. Αξιοσημείωτο είναι επίσης το γεγονός ότι η υλοποίηση με FFT, είναι πιο αργή από την απλή, αλλά ολικά vectorized, συνέλιξη. Για τον υπολογισμό της πολυπλοκότητας χρησιμοποιήθηκαν οι γενικοί τύποι (δηλαδή όχι οι αναλυτικοί), που προέκυψαν από την θεωρητική ανάλυση στα ζητούμενα 3.1 και 3.2.