Επιστημονικός Υπολογισμός 1

Τμήμα Μηχανικών Η/Υ και Πληροφορικής

Πανεπιστήμιο Πατρών

2η Εργαστηριακή Άσκηση

Created with LaTeX

Δούκας Ιωάννης 5509

Περιεχόμενα

Χαρακτηριστικά Υπολογιστικού Συστήματος

Τα χαρακτηριστικά του υπολογιστικού συστήματος στο οποίο υλοποιήθηκαν οι ασκήσεις στο περιβάλλον matlab περιγράφονται στον παρακάτω πίνακα.

Model Name:	MacBook Pro 9.2
Processor:	Intel(R) Core(TM) i5-3210M 2.50GHz (dual core)
L2 Cache (per Core):	256 KB
L3 Cache:	3 MB
Merory (Ram):	4 GB
Operating System:	OS X 10.9.5
Matlab Version:	8.3.0.532 (R2014a) 64-bit
Διακριτότητα Χρονομετρητή:	2.9254e-07
Αποτέλεσμα LU από εντολή bench:	0.1141
FMA instruction set	Δεν υποστηρίζεται

Παρατήρηση: Η διακριτότητα του χρόνου βρέθηκε εκτελώντας το παρακάτω script.

```
nmax = 100:100:100000;
t=0;
a = 0; b=0; c=0;
for i=0:length(nmax)
    tic;
    a = b+c;
    t = t + toc;
end
t = t/length(nmax)
```

1 Πράξεις με Πολυώνυμα

Σκοπός αυτής της παραγράφου είναι η βαθύτερη κατανόηση της αριθμητικής πεπερασμένης ακρίβειας την οποία χρησιμοποιεί η Matlab. Θα διερευνήσουμε την επίδραση της αριθμητικής πεπερασμένης ακρίβειας πάνω σε πράξεις με πολυώνυμα. Για να το κάνουμε όμως αυτό, αρχικά, θα πρέπει να μελετήσουμε ορισμένες συναρτήσεις της Matlab οι οποίες είναι σχεδιασμένες για να τη διαχείριση πολυωνύμων.

1.1 Συναρτήσεις poly και polyval

Δύο συναρτήσεις της matlab οι οποίες είναι σχεδιασμένες για τη διαχείριση πολυωνύμων είναι η *poly* και η *polyval*. Η συνάρτηση *poly* αυτό το οποίο κάνει είναι δοσμένου ενός μητρώου A διαστάσεων $n \times n$ επιστρέφει ένα διάνυσμα γραμμή, $n+1$ στοιχείων, το οποίο αντιστοιχεί στο χαρακτηριστικό πολυώνυμο του μητρώου A. Το χαρακτηριστικό πολυώνυμο είναι το αριστερό μέρος της χαρακτηριστικής εξίσωσης του μητρώου: $det(A-\lambda x) = 0$. Αυτό το οποίο κάνει είναι να υπολογίζει στην αρχή τις ιδιοτιμές και με βάση αυτές δημιουργεί το πολυώνυμο "εισάγοντας" του μία μία τις ρίζες τις οποίες θέλουμε. Στην περίπτωση που το είναι διάνυσμα αυτό το οποίο κάνει είναι να δημιουργεί ένα πολυώνυμο σε μορφή δυναμορφής με ρίζες τις τιμές του διανύσματος A . Και στις δύο περιπτώσεις αυτό όμως δεν είναι απόλυτα σωστό σαν πρόγραμμα καθώς οι συντελεστές επηρεάζονται από τα σφάλματα στρογγύλευσης του μητρώου A. Η συνάρτηση *polyval* αυτό το οποίο κάνει είναι να υπολογίζει την τιμή ενός πολυωνύμου $p(x)$ για τα x τα οποία δίνονται σαν είσοδο. Επίσης μπορεί εκτός από την πολυωνύμου να επιστρέψει και μία τιμή $delta$ η οποία το σφάλμα του υπολογισμού της τιμής. Με την εντολή *edit polyval.m* στη matlab μπορούμε να διαβάσουμε τον κώδικα της. Βλέπουμε πως η *polyval* στην γενική περίπτωση εφαρμόζει αλγόριθμο Horner. Στη συνέχεια ανάλογα με το πόσα ορίσματα πρέπει να επιστρέψει κατασκευάζει, αν ζητάτε, το μητρώο Vardermonde και το υπολογίζει το σφάλμα των υπολογισμών.

1.2 Πολυώνυμα της μορφής $\prod_{i=0}^{n}(x-i)$

Σε αυτό το σημείο θα μελετήσουμε τα πολυώνυμα της μορφής $\prod_{i=0}^{n}(x-i)$ τα οποία μπορεί να φαίνονται απλά, αλλά κρύβουν παγίδες. Για τη μελέτη υλοποιήσαμε το παρακάτω script.

```
1 n = 15:2:25;
2 x_1 = zeros( 1,length(n) );
3 x_n = zeros( 1,length(n) );
4 colors = ['yo', 'm*', 'cx', 'rd', 'gV', 'bp','kh'];
5 c = 1:2;
6 h = figure;
```

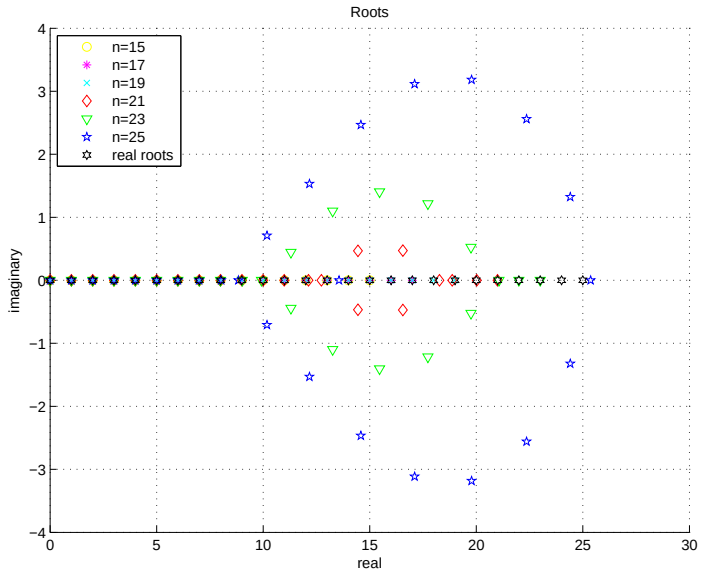
```
grid on
hold on
for i=1:length(n),
    a_n = poly(0:n(i));
    x_1(i) = polyval( a_n, 1 );
    x_n(i) = polyval( a_n, n(i) );
    r = roots( a_n );

    plot( real(r), imag(r), colors( c ));
    c = c+2;
    clear a_n r

end
plot(0:n(i), zeros(n(i)+1,1), colors( c ));
legend('n=15','n=17','n=19','n=21','n=23','n=25', 'real ...
    roots','Location','NorthWest');
xlabel(' real ');
ylabel(' imaginary ');
title(' Roots ');
x_1
x_n
hold off
%saveas(h,'askplot3','eps')
```

script 1.b

Η έξοδος του script φαίνεται παρακάτω.

i	x=1	x=n
15	0	0
17	0	0
19	0	633881344
21	-65536	481754825588736
23	8388608	8,38318958974494e+18
25	1.5032e+10	-4,28076327862722e+23

Παρατηρούμε ότι καθώς μεγαλώνει το n το σφάλμα μεγαλώνει τόσο στον υπολογισμό των $p(1), p(x)$ όσο και στον υπολογισμό των ριζών του πολυωνύμου. Πιο αναλυτικά στον υπολογισμό των $p(1), p(n)$ θα περιμέναμε κάθε φορά το αποτέλεσμα να είναι 0 καθώς το πολυώνυμο που κατασκευάζουμε απαιτεί να τα έχει ρίζες. Βλέπουμε όμως ότι για $n > 19$ έχουμε $p(n) = 633881344$ που σίγουρα απέχει αρκετά από το 0 που θέλουμε εμείς. Αυτό συμβαίνει γιατί καθώς κάνουμε πράξεις για τον υπολογισμό των συντελεστών το σφάλμα μεγαλώνει μέχρι που φτάνει σε ένα σημείο που τα αποτελέσματα είναι τελείως λάθος. Αντίστοιχα και για τις υπόλοιπες τιμές. Στην συνέχεια βλέπουμε την γραφική παράσταση με τις ρίζες του πολυωνύμου που κατασκεύασε. Παρατηρούμε ότι, καθώς μεγαλώνει το n όχι μόνο κάποιες ρίζες απέχουν από τις πραγματικές αλλά είναι και μιγαδικές.



Σχήμα 1.1 : ρίζες πολυωνύμου $\prod_{i=0}^{n}(x-i)$ από roots της Matlab

2 Αθροίσματα - Μονάδα Στρογγύλευσης

Η άθροιση μεγάλου συνόλου αριθμών μπορεί να οδηγήσει σε σημαντικά σφάλματα στρογγύλευσης. Υπάρχουν διάφορες μέθοδοι για να ελεγχθεί αυτό. Παρακάτω θα εξετάσουμε μερικές από αυτές.

2.1 Υπολογισμός αθροίσματος μονής ακρίβειας

Για την μελέτη των σφαλμάτων στρογγύλευσης σε αυτό το σημείο υλοποιήσαμε ένα script σε Matlab στο οποίο χρησιμοποιούμε διαφορετικές αριθμητικές μεθόδους για την πρόσθεση ενός συνόλου x αριθμών μονής ακρίβειας. Παρακάτω φαίνεται το αντίστοιχο script.

```
1  x = single(x);
2
3  fprintf('--- apo aristera deksia --- \n');
4  sum = single( x(1) );
5  for i=2:length(x),
6      sum = sum + single( x(i) );
7  end
8  disp(sum);
9
10 fprintf('--- taksinomimeno --- \n');
```

```
x_sort = single( sort(x) );
sum_s = single( x_sort(1) );
for i=2:length(x_sort),
    sum_s = sum_s + x_sort(i);
end
disp(sum_s);

fprintf('--- epanataksinomimeno --- \n');
x_ss = single( x_sort )';
for i=1:length(x)-1,
    sum_ss = x_ss(1) + x_ss(2);

    x_ss(2) = sum_ss;
    x_ss = x_ss(2:length(x_ss));
    x_ss = sort( x_ss );

end
disp(sum_ss)

fprintf('--- Pichat & Neumaier  --- \n');
s = single(x(1));
e = 0;
for i=2:length(x),
    if abs(s) ≥ abs(x(i))
        [s,e_i] = fast2sum(s, x(i));
    else
        [s,e_i] = fast2sum(x(i), s);
    end
    e = e + e_i;
end
s = s + e;
disp(s)
```

script 2

Το παραπάνω script αυτό το οποίο κάνει είναι να παίρνει ένα διάνυσμα x σαν είσοδο και να υπολογίζει το άθροισμα των στοιχείων του με 4 διαφορετικούς τρόπους. Το παραπάνω script το ελέγξαμε καλώντας το για διάφορα x τα οποία φαίνονται στο επόμενο script.

```
format LONGG
x = single( zeros(64,1) );

% i
fprintf('\n\n---------- taylor ------------ \n\n');
for i=0:63,
    x(i+1) = ( -2*pi )^i /  factorial(i);
end
ask2

```

```
x = single( zeros(4096,1) );
% ii
fprintf('\n\n-------- 1, 10e-18, -1 -------- \n\n');
x(1:2047) = single( ones(2047,1) );
x(2048:2049) = single( 1e-18 );
x(2050:4096) = single( -1 );
ask2

% iii
fprintf('\n\n------------ [1,2] ------------ \n\n');
x = single( linspace(1,2,4096) );
ask2

% iv
fprintf('\n\n----------- 1/(i^2) ----------- \n\n');
for i=1:4096,
    x(i) = single( 1/(i^2) );
end
ask2
format short
```

script test_ask2

Η έξοδος του παραπάνω script φαίνεται στη συνέχεια.

	αριστερά δεξιά	ταξινομημένο	επαναταξινομημένο
Taylor: $e^{-2\pi}$	0.001868146	0.001968384	0.001968384
1, 10e-18, -1	0	0	0
$[1, 2]$	6143.982	6143.982	6144
$1/i^2$	1.644725	1.64469	1.64469

	Pitchat and Neumaier's	πραγματικά αποτελέσματα
Taylor: $e^{-2\pi}$	0.001968384	0.00186744273171767
1, 10e-18, -1	2e-18	2e-18
$[1, 2]$	6144	6144
$1/i^2$	1.64469	1.64468995602312

Οι αριθμοί αναπαριστάνονται στο πρότυπο της IEEE 754, στο οποίο ένας αριθμός έχει παραπάνω από μία αναπαραστάσεις στο δυαδικό. Για αυτό το λόγο οι αριθμοί αποθηκεύονται κανονικοποιημένοι. Οπότε όταν θέλουμε να προσθέσουμε δύο νούμερα αυτό που έχει το μικρότερο εκθέτη αλλάζει και παίρνει τον εκθέτη του μεγαλύτερου. Αυτό έχει σαν αποτέλεσμα να χάνονται μερικά από τα λιγότερο σημαντικά του ψηφία. Συγκρίνοντας τα πραγματικά αποτελέσματα με αυτά που υπολογίσαμε βλέπουμε πως η μέθοδος η οποία ανταποκρίνεται καλύτερα από όλες είναι πρώτη η Pitchat and Neumaier's, δεύτερη η μέθοδος που τα επαναταξινομεί, τρίτη αυτή που ταξινομεί και τελευταία η μέθοδος που απλά τα προσθέτει από

αριστερά προς τα δεξιά. Αυτό είναι αναμενόμενο καθώς η μέθοδος που προσθέτει τους αριθμούς αφού πρώτα τους έχει ταξινομήσει έχει μικρότερο σφάλμα, γιατί δεν έχουμε πολλές προσθέσεις με αριθμούς με διαφορετικό εκθέτη. Η βελτιστοποιημένη μέθοδος αυτής της ιδέας είναι να επαναξινομούμε και το άθροισμα το οποίο παίρνουμε σαν αποτέλεσμα γιατί το άθροισμα καθώς γίνονται οι προσθέσεις αυξάνεται σαν νούμερο. Τέλος η μέθοδος Pitchat and Neumaier's έχει το μικρότερο σφάλμα από όλες γιατί αυτό το οποίο κάνει είναι επειδή γνωρίζουμε ότι έχουμε αυτό το σφάλμα αυτό, το υπολογίζει και το προσθέτει στο αποτέλεσμα που παίρνουμε.

3 Γραμμικά Συστήματα

3.1 Αξιολόγηση σφαλμάτων α.κ.υ. επίλυσης $Ax = b$

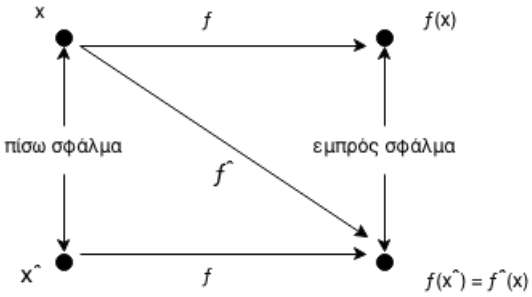
Σε αυτό το σημείο θα αξιολογήσουμε σφάλματα αριθμητικής κινητής υποδιαστολής που μπορεί να προκύψουν για την πράξη επίλυσης συστήματος εξισώσεων $Ax = b$, όπου $A \in \mathbb{R}^{n\times n}, x \in \mathbb{R}^{n\times 1}, b \in \mathbb{R}^{n\times 1}$.

```
function [cn, e_sx_sf, b_sf, b_c, db_b, x_prog] = ask3a(A)
% ASK3
% INPUT
%   A           - matrix
% OUTPUT
%   cn          - condition number
%   e_sx_sf     - forward error
%   b_sf        - backward error with norm infinity
%   b_c         - backward error
%   db_b        - || b^ - b || / || b ||

    n = length(A);
    cn = cond(A, inf);

    x = ones(n,1);
    b = A*x;

    x_prog = A \ b;

    % forward error
    e_sx_sf = norm(x_prog - x, inf) / norm(x, inf);

    % backward error with norm inf
    b_sf = norm( A*x_prog - b , inf) / (norm( A , inf) * ...
        norm( x_prog , inf) + norm( b , inf));

    % backward error
    b_c = norm( A*x_prog - b , inf ) / (norm( A , inf) * ...
        norm( x_prog , 1) + norm( b , inf));
```

```
28
29      db_b = norm( A*x_prog - b, inf ) / norm( b, inf );
30  end
```

Ερώτημα 3α

Ο δείκτης κατάστασης ενός μητρώου, το εμπρός σφάλμα και το πίσω σφάλμα έχουν να κάνουν με την ευαισθησία της διατάραξης του προβλήματος, που καλούμαστε να λύσουμε, το καθένα όμως με το δικό του ρόλο. Ο δείκτης κατάστασης του μητρώου A μας λέει πόσο ευαίσθητο είναι το εν λόγο μητρώο. Στην περίπτωση που το μητρώο είναι κακό, δηλαδή έχει δείκτη κατάστασης $>> 1$, τότε το πρόβλημα δεν μπορεί υπολογιστικά να λυθεί. Το εμπρός σχετικό σφάλμα μας λέει κατά πόσο διαταράσσεται η λύση του προβλήματος μας, δηλαδή κατά πόσο η λύση η οποία πήραμε είναι κοντά στην λύση την οποία ψάχνουμε εμείς να βρούμε. Το πίσω σφάλμα είναι το μέτρο το οποίο μας λέει πόσο απέχει η είσοδος που πρέπει να δώσουμε ώστε να πάρουμε σαν λύση την προσεγγιστική λύση που παίρνουμε για την πραγματική είσοδο.



3.2 Θεωρητικές προβλέπεις

Δεδομένου των σφαλμάτων στρογγύλευσης που γίνονται στον υπολογιστή όταν εμείς θέλουμε για ένα δοσμένο μητρώο A να λύσουμε το γραμμικό σύστημα $Ax = b$ γνωρίζουμε ότι στην πραγματικότητα λύνουμε το σύστημα $A\hat{x} = b + \Delta b$ όπου $\hat{x}$ η λύση η οποία υπολογίζουμε. Ορίζοντας την διαφορά $\Delta x = \hat{x} - x$ παίρνουμε ότι:

$$b + \Delta b = A\hat{x} = A(x + \Delta x) = Ax + A\Delta x \tag{3.2.1}$$

Γνωρίζοντας ότι θεωρητικά ισχύει $Ax = b$ θα πρέπει να έχουμε $A\Delta x = \Delta b \Rightarrow \Delta x = A^{-1}\Delta b$. Οπότε έχουμε ότι:

$$b = Ax \Rightarrow \|b\| \leq \|A\| \, \|x\| \tag{3.2.2}$$

και

$$\Delta x = A^{-1}\Delta b \Rightarrow \|\Delta x\| \leq \left\|A^{-1}\right\| \|\Delta b\| \tag{3.2.3}$$

και από τον ορισμό του δείκτη κατάστασης του μητρώου A, $cond(A) = \left\|A^{-1}\right\| \|A\|$ έχουμε ότι:

$$\frac{\|\Delta x\|}{\|x\|} \leq cond(A)\frac{\|\Delta b\|}{\|b\|} \tag{3.2.4}$$

και επίσης γνωρίζουμε ότι ισχύει η ανισότητα:

$$\frac{\|\Delta x\|}{\|x\|} \leq cond(A)\beta_N \tag{3.2.5}$$

όπου β_N είναι η εκτίμηση του πίσω σφάλματος. Οπότε το εμπρός σχετικό φράγμα για την επίλυση γραμμικού συστήματος φράσσετε όπως φαίνεται στις εξισώσεις 3.2.4 και 3.2.5.

Όσο αναφορά το πίσω σφάλμα γνωρίζουμε ότι μπορούμε να το εκτιμήσουμε εύκολα από τον τύπο:

$$\beta_N = \frac{\|b - A\hat{x}\|_\infty}{\|A\|_\infty \|\hat{x}\|_\infty + \|b\|_\infty} \tag{3.2.6}$$

Ένας άλλος εναλλακτικός τρόπος για την εκτίμηση του πίσω σφάλματος έχει δοθεί από τον Nicholas J. Higham ο οποίος είναι παρόμοιος αλλά πιο αυστηρός. Για αρχή θα πρέπει να ορίσουμε ένα μητρώο $E \geq 0$ και ένα διάνυσμα $f \geq 0$ των οποίων τα στοιχεία παρέχουν μία σχετική ανοχή κατά των στοιχείων των ΔA και Δb. Για το διάνυσμα $\hat{x}$ το πίσω σφάλμα ορίζεται:

$$\beta_C(E, f) = \min\{\omega : (A + \Delta A)\hat{x} = b + \Delta b, |\Delta A| \leq \omega E, |\Delta b| \leq \omega f, \\ \Delta A \in \mathbb{R}^{n\times n}, \Delta b \in \mathbb{R}^n\} \tag{3.2.7}$$

Για την παραγοντοποίηση LU ορίζεται ως: $\beta_C(E) = \max\{|\Delta a_{ij}|/e_{ij}, 1 \leq i, j \leq n\}$. Δύο ακραίες επιλογές για το μητρώο E και για το διάνυσμα f είναι:

1. $E = |A|$: Σε αυτή την περίπτωση μετράμε το μέγεθος των διαταράξεων για κάθε στοιχείο A ή του b σε σχέση με κάθε στοιχείο του εαυτού του. Αυτό σύμφωνα με τον Nicholas J. Higam είναι το πιο αυστηρό πίσω σφάλμα για ένα μητρώο χωρίς ειδική δομή.

2. $E = \|A\|_\infty ee^T, f = \|b\|_\infty e, e = (1, 1, \ldots, 1)^T$: Σε αυτή την περίπτωση μετράμε τη διατάραξη σε απόλυτη τιμή, παρόμοια με την εξίσωση 3.2.6 αλλά όχι ίδια.

Μία έκφραση για το β_C έχει δοθεί από τον Oettli και Prager στην οποία έχουμε ότι:

$$\beta_C = \max_i \frac{|b - A\hat{x}|_i}{(E|\hat{x}| + f)_i} \tag{3.2.8}$$

και όταν έχουμε $0/0$ παίρνουμε 0 και όταν έχουμε κάποιο νούμερο προς 0 παίρνουμε άπειρο. Για τον υπολογισμό αυτού του σφάλματος για την δεύτερη ακραία περίπτωση έχουμε:

$$\beta_C = \frac{\|b - A\hat{x}\|_\infty}{\|A\|_\infty \|\hat{x}\|_1 + \|b\|_\infty} \tag{3.2.9}$$

Για την επιβεβαίωση των θεωρητικών προβλέψεων των σφαλμάτων έχουμε το παρακάτω script και η έξοδος του φαίνεται στον πίνακα που το ακολουθεί.

```
clc
n = 512;

fprintf('eps = ');
disp(eps);

% random matrix A
fprintf('\n------------ random matrix A ------------\n');
A = randn(n);
[cn, e_sx_sf, b_sf, b_c, db_b, f_prog_x] = ask3a(A);
fprintf('condition number            :');
disp(cn);
fprintf('forward error               :');
disp(e_sx_sf);
fprintf('backward error (norm inf)   :');
disp(b_sf);
fprintf('backward error (norm inf,1) :');
disp(b_c);
bound1 = cn*b_sf;
bound2 = cn*b_c;
fprintf('bound1: %d ≤ %d \n', e_sx_sf, bound1);
fprintf('bound2: %d ≤ %d \n', e_sx_sf, bound2);
fprintf('bound3: %d ≤ %d \n', e_sx_sf, db_b);

% lower triangular matrix
fprintf('\n--------- lower triangular matrix ---------\n');
A = tril( randn(n) );
[cn, e_sx_sf, b_sf, b_c, db_b, f_prog_x] = ask3a(A);
fprintf('condition number            :');
disp(cn);
fprintf('forward error               :');
disp(e_sx_sf);
fprintf('backward error (norm inf)   :');
disp(b_sf);
fprintf('backward error (norm inf,1) :');
disp(b_c);
bound1 = cn*b_sf;
bound2 = cn*b_c;
fprintf('bound1: %d ≤ %d \n', e_sx_sf, bound1);
fprintf('bound2: %d ≤ %d \n', e_sx_sf, bound2);
fprintf('bound3: %d ≤ %d \n', e_sx_sf, db_b);
% lu
fprintf('\n------------------- lu --------------------\n');
[ans,A] = lu( randn(n) );
[cn, e_sx_sf, b_sf, b_c, db_b, f_prog_x] = ask3a(A);
fprintf('condition number            :');
disp(cn);
fprintf('forward error               :');
disp(e_sx_sf);
```

```
50 fprintf('backward error (norm inf)   :');
51 disp(b_sf);
52 fprintf('backward error (norm inf,1) :');
53 disp(b_c);
54 bound1 = cn*b_sf;
55 bound2 = cn*b_c;
56 fprintf('bound1: %d ≤ %d \n', e_sx_sf, bound1);
57 fprintf('bound2: %d ≤ %d \n', e_sx_sf, bound2);
58 fprintf('bound3: %d ≤ %d \n', e_sx_sf, db_b);
59
60 % gfpp
61 fprintf('\n------------------ gfpp -------------------\n');
62 temp = triu( randn(n-1) );
63 A = gfpp(temp);
64 [cn, e_sx_sf, b_sf, b_c, db_b, f_prog_x] = ask3a(A);
65 fprintf('condition number            :');
66 disp(cn);
67 fprintf('forward error               :');
68 disp(e_sx_sf);
69 fprintf('backward error (norm inf)   :');
70 disp(b_sf);
71 fprintf('backward error (norm inf,1) :');
72 disp(b_c);
73 bound1 = cn*b_sf;
74 bound2 = cn*b_c;
75 fprintf('bound1: %d ≤ %d \n', e_sx_sf, bound1);
76 fprintf('bound2: %d ≤ %d \n', e_sx_sf, bound2);
77 fprintf('bound3: %d ≤ %d \n', e_sx_sf, db_b);
78
79 % vardermonde [-1,1]
80 fprintf('\n----------- vardermonde [-1,1] ------------\n');
81 A = vander( linspace(-1,1,n) );
82 [cn, e_sx_sf, b_sf, b_c, db_b, f_prog_x] = ask3a(A);
83 fprintf('condition number            :');
84 disp(cn);
85 fprintf('forward error               :');
86 disp(e_sx_sf);
87 fprintf('backward error (norm inf)   :');
88 disp(b_sf);
89 fprintf('backward error (norm inf,1) :');
90 disp(b_c);
91 bound1 = cn*b_sf;
92 bound2 = cn*b_c;
93 fprintf('bound1: %d ≤ %d \n', e_sx_sf, bound1);
94 fprintf('bound2: %d ≤ %d \n', e_sx_sf, bound2);
95 fprintf('bound3: %d ≤ %d \n', e_sx_sf, db_b);
96
97 % vardermonde cos(k\pi/(n+1) ,k=0:n)
98 fprintf('\n------- vardermonde cos(k*pi/(n+1) --------\n');
99 k = 0:n;
100 A = vander( cos(k*pi/(n+1)) );
101 [cn, e_sx_sf, b_sf, b_c, db_b, f_prog_x] = ask3a(A);
102 fprintf('condition number            :');
103 disp(cn);
```

```
fprintf('forward error                    :');
disp(e_sx_sf);
fprintf('backward error (norm inf)    :');
disp(b_sf);
fprintf('backward error (norm inf,1) :');
disp(b_c);
bound1 = cn*b_sf;
bound2 = cn*b_c;
fprintf('bound1: %d ≤ %d \n', e_sx_sf, bound1);
fprintf('bound2: %d ≤ %d \n', e_sx_sf, bound2);
fprintf('bound3: %d ≤ %d \n', e_sx_sf, db_b);
```

script 3a

	Δείκτης κατ.	Εμπρός Σφ.	Πίσω σφ. i	Πίσω σφ. ii
τυχαίο	9.1020e+04	6.5592e-13	2.0228e-15	4.6755e-18
κάτω τριγωνικό	1.302435e-14	1.0790e+136	1.0790e+136	2.0574e-25
από LU	2.9677e+04	2.9677e+04	7.1991e-17	1.6300e-19
gfpp	4.6156e+155	3.4318e+138	1.3252e-21	1.3252e-21
vardermonde $[-1,1]$	4.8990e+222	3.5278e+203	9.6677e-20	9.6677e-20
vardermonde $\cos(\frac{k\pi}{n+1})$	7.4152e+154	9.1397e+135	2.1052e-18	2.1052e-18

Πίνακας 1:

	Φράγμα i	Φράγμα ii	Φράγμα iii
τυχαίο	1.841152e-10	4.255652e-13	1.302435e-14
κάτω τριγωνικό	5.339525e-05	2.032620e-05	2.032620e-05
από LU	2.136511e-12	4.837557e-15	4.837557e-15
gfpp	6.116443e+134	5.758852e+134	5.602412e+118
vardermonde $[-1,1]$	4.736219e+203	4.423593e+202	3.410567e+184
vardermonde $\cos(\frac{k\pi}{n+1})$	1.561079e+137	5.967637e+135	1.924141e+118

Πίνακας 2:

Παρατηρούμε ότι τα φράγματα που γνωρίζουμε δεν φαίνονται όπως τα υπολογίζουμε να ισχύουν. Αν προσέξουμε καλύτερα θα δούμε ότι η μεγαλύτερη απόκλιση υπάρχει όταν ο δείκτης κατάστασης του μητρώου είναι πολύ μεγάλος, δηλαδή το μητρώο είναι πολύ κακό. Οπότε είναι φυσιολογικό όλα τα μεγέθη που υπολογίζουμε για αυτά τα μητρώα να απέχουν αρκετά από τα πραγματικά και θα πρέπει να περιμένουμε ότι τα περισσότερα αποτελέσματα δεν θα είναι τα αναμενόμενα.

3.3 Αξιολόγηση σφαλμάτων α.κ.υ. πολλαπλασιασμού μητρώων

Σε αυτό το σημείο συγκρίνουμε σφάλματα αριθμητικής κινητής υποδιαστολής που προκύπτουν για την πράξη του κλασικού πολλαπλασιασμού μητρώων και τον

block πολλαπλασιασμού μητρώων χρησιμοποιώντας τον αλγόριθμο *Strassen*. Τη σύγκριση την κάνουμε με την matlab για την συνάρτηση *mtimes* και τη συνάρτηση *strassen*.

```
function [m_f_e, s_f_e] = ask3b(A,B)
% ASK3B
% INPUT
%  A          - First matrix for multiply
%  B          - Second matrix for multiply
% OUTPUT
% m_f_e       - Mtimes forward error with norm inf
% s_f_e       - Strassen forward error with nomr inf
    f_x = A*B;
    f_x_prog = mtimes(single(A), single(B));
    m_f_e = norm( f_x - f_x_prog, inf ) / norm( f_x, inf )

    f_x_prog = strassen(single(A), single(B));
    s_f_e = norm( f_x - f_x_prog, inf ) / norm( f_x, inf )
end
```

Ερώτημα 3b

Για την σύγκριση των δύο αλγορίθμων εκτελέσαμε το παρακάτω script:

```
n = 1024;

fprintf('---------- Random matrix ----------\n');
ask3b(rand(n,n), rand(n,n));

fprintf('------- Vandremonde matrix --------\n');
ask3b(vander( rand(n,1) ), vander( rand(n,1) ) );

fprintf('------- telefteo matrix --------\n');
n = 512;
M = 10^7;
A = [eye(n), zeros(n); zeros(n), eye(n)] * [M*rand(n), ...
    rand(n); rand(n), rand(n)];
B = [eye(n), zeros(n); zeros(n), eye(n)] * [M*rand(n), ...
    rand(n); rand(n), rand(n)];
ask3b(A,B);
```

script 3b

Η έξοδος του script φαίνεται στον παρακάτω πίνακα:

	mtimes	Strassen
(i)	1.0327e-07	1.6643e-06
(ii)	7.8233e-08	3.5150e-06
(iii)	1.4832e-07	1.4318e-06

Παρατηρούμε ότι για κάθε τύπο μητρώου η συνάρτηση mtimes της matlab έχει πολύ καλύτερη ακρίβεια σε σχέση με τη Strassen όταν εκτελείται με την default πλοκαδοποίηση ($= 8$).

Βιβλιογραφία

[1] . Gilbert Strang. Εισαγωγή στη γραμμική άλγεβρα. Εκδόσεις Πανεπιστήμιο Πατρών 2012

[2] . Επιστημονικός υπολογισμός Ι. Ευστράτιος Γαλλόπουλος. 2013

[3] . *http://www.mathworks.com/help/matlab/*

[4] . Scientific Computing:An Introductory Survery. Michael T. Heath

[5] . How Accurate is Gaussian Elimination?. Nicholas J. Higham, Department of Computer Science Cornell University Ithaca, New York 14853